Department of Electrical and Information Engineering

ELECTRICAL AND INFORMATION ENGINEERING PH.D. PROGRAM

SSD: ING-INF/05 - INFORMATION PROCESSING SYSTEMS

Final Dissertation

Graph Neural Networks for Recommendation leveraging Multimodal Information

by

Daniele Malitesta

Supervisor

Prof. Tommaso Di Noia

Coordinator of the Ph.D. Program

Prof. Mario Carpentieri

Course XXXVI, 01/11/2020 - 31/10/2023

Department of Electrical and Information Engineering

ELECTRICAL AND INFORMATION ENGINEERING PH.D. PROGRAM

SSD: ING-INF/05 - INFORMATION PROCESSING SYSTEMS

Final Dissertation

Graph Neural Networks for Recommendation leveraging Multimodal Information

by

Daniele Malitesta

[illegible]

Referees

Prof. Ludovico Boratto

Prof. Pasquale Minervini

Supervisor

Prof. Tommaso Di Noia

[illegible]

Coordinator of the Ph.D. Program

Prof. Mario Carpentieri

Mario Carpentieri

Course XXXVI, 01/11/2020 - 31/10/2023

Dedicated to those I love.

Abstract

In the era of digital information overload on the Internet, recommender systems act as filtering algorithms to provide users with items that might meet their interests according to expressed preferences and items' properties and characteristics. Among the various recommendation paradigms so far, collaborative filtering has represented the most successful one thanks to the easy application of its recommendation algorithms whose high level performance in terms of recommendation accuracy is widely recognized. Despite the recent success of machine and deep learning techniques for collaborative filtering which have been effectively applied to recommender systems to improve the learning and profiling abilities of such models, recommendation still remains an highly-challenging task. Among the most debated open issues in the community, this thesis considers two algorithmic and conceptual ones, namely: (i) the inexplicable nature of users' preferences, especially when they come in the form of implicit feedback; (ii) the effective exploitation of the collaborative information in the designing and training of recommendation models.

In specific scenarios and domains such as fashion, food, tourism, and media content recommendation, the shallow item's profile, commonly based upon the information conveyed within the user-item interactions only, may be enhanced through the multi-faceted characteristics describing items. Driven from these assumptions, in the first part of this thesis, we propose to apply multimodal deep learning strategies for multimedia recommendation; the scope is to study and design recommendation algorithms based upon the principles of multimodality to possibly match each items' characteristic to the implicit preference expressed by the user, thus addressing the (i) issue.

Recent collaborative filtering approaches leverage the representational power of machine learning systems to profile users and items through embedding vectors in the latent space. In doing so, however, such recommendation models disregard a wide range of structural properties which are naturally encoded into the user-item interaction data. Indeed, recommendation datasets are easily describable under the topology of a bipartite and undirected graph, with users and items being the graph nodes connected at multiple distance hops. In this respect, the application of graph neural networks,

recent machine learning techniques specifically tailored to learn from non-euclidean data, is of the utmost importance to provide a refined representation of users and items which can mine near- and long-distance relationships in the user-item graphs. Indeed, this is one possible way to exploit the collaborative signal, which is effectively propagated within the user-item graph, thus addressing the (ii) issue.

Far from considering multimodal-aware and graph-based recommender systems from a separate perspective, this thesis conclusively aims to match the two families of recommendation strategies to propose an approach leveraging graph neural networks and multimodal information data. In seeking this joint research objective, other numerous micro aspects within the two macro areas (introduced above) are examined. Indeed, the thesis is a systematic compendium of careful additional analyses regarding, among the others, reproducibility, novel evaluation dimensions, and tasks and scenarios complementary to recommendation.

Publications

Some ideas and figures have appeared previously in other publications. A complete list of my publications during the PhD is available in the following. Note that, in some cases, the author list follows the alphabetical order. For this reason, for each paper, corresponding authors are explicitly denoted through **boldface**.

[1] **Alberto Carlo Maria Mancino**, **Antonio Ferrara**, **Salvatore Bufi**, Daniele Malitesta, Tommaso Di Noia, and Eugenio Di Sciascio. “KGTORe: Tailored Recommendations through Knowledge-aware GNN Models.” In: *RecSys*. ACM, 2023, pp. 576–587.

[2] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. “V-Elliot: Design, Evaluate and Tune Visual Recommender Systems.” In: *RecSys*. ACM, 2021, pp. 768–771.

[3] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, **Daniele Malitesta**, and **Felice Antonio Merra**. “A Study of Defensive Methods to Protect Visual Recommendation Against Adversarial Manipulation of Images.” In: *SIGIR*. ACM, 2021, pp. 1094–1103.

[4] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, **Daniele Malitesta**, Vincenzo Paparella, and **Claudio Pomo**. “Auditing Consumer- and Producer-Fairness in Graph Collaborative Filtering.” In: *ECIR (1)*. Vol. 13980. Lecture Notes in Computer Science. Springer, 2023, pp. 33–48.

[5] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, **Daniele Malitesta**, and **Claudio Pomo**. “How Neighborhood Exploration influences Novelty and Diversity in Graph Collaborative Filtering.” In: *MORS@RecSys*. Vol. 3268. CEUR Workshop Proceedings. CEUR-WS.org, 2022.

[6] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, **Daniele Malitesta**, and **Claudio Pomo**. "Reshaping Graph Recommendation with Edge Graph Collaborative Filtering and Customer Reviews." In: *DL4SR@CIKM*. Vol. 3317. CEUR Workshop Proceedings. CEUR-WS.org, 2022.

[7] Vito Walter Anelli, Tommaso Di Noia, **Daniele Malitesta**, and **Felice Antonio Merra**. "Assessing Perceptual and Recommendation Mutation of Adversarially-Poisoned Visual Recommenders (short paper)." In: *DP@AI*IA*. Vol. 2776. CEUR Workshop Proceedings. CEUR-WS.org, 2020, pp. 49–56.

[8] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Daniele Malitesta, and **Felice Antonio Merra**. "Adversarial Attacks against Visual Recommendation: an Investigation on the Influence of Items' Popularity." In: *OHARS@RecSys*. Vol. 3012. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 33–44.

[9] Ludovico Boratto, Daniele Malitesta, Mirko Marras, Giacomo Medda, Cataldo Musto, and Erasmo Purificato. "First International Workshop on Graph-Based Approaches in Information Retrieval (IRonGraphs 2024)." In: *To Appear in Proceedings of the 46th European Conference on Information Retrieval* (2024).

[10] **Daniele Malitesta**, **Claudio Pomo**, Vito Walter Anelli, Alberto Carlo Maria Mancino, Eugenio Di Sciascio, and Tommaso Di Noia. "A Topology-aware Analysis of Graph Collaborative Filtering." In: arXiv:2308.10778 (2023).

[11] **Daniele Malitesta**, **Claudio Pomo**, Vito Walter Anelli, Tommaso Di Noia, and Antonio Ferrara. "An Out-of-the-Box Application for Reproducible Graph Collaborative Filtering extending the Elliot Framework." In: *UMAP (Adjunct Publication)*. ACM, 2023, pp. 12–15.

[12] **Daniele Malitesta**, **Claudio Pomo**, and Tommaso Di Noia. "Graph Neural Networks for Recommendation: Reproducibility, Graph Topology, and Node Representation." In: *Accepted as tutorial at the 2nd Learning on Graphs Conference* arXiv:2310.11270 (2023).

[13] **Daniele Malitesta**, Giandomenico Cornacchia, Claudio Pomo, Felice Antonio Merra, Tommaso Di Noia, and Eugenio Di Sciascio. "Formalizing Multimedia Recommendation through Multimodal Deep Learning." In: *Under review at ACM Transactions on Recommender Systems* arXiv:2309.05273 (2023).

[14] **Daniele Malitesta**, **Giandomenico Cornacchia**, Claudio Pomo, and Tommaso Di Noia. "Disentangling the Performance Puzzle of Multimodal-aware Recommender Systems." In: *EvalRS@KDD*. Vol. 3450. CEUR Workshop Proceedings. CEUR-WS.org, 2023.

[15] **Daniele Malitesta**, **Giandomenico Cornacchia**, Claudio Pomo, and Tommaso Di Noia. “On Popularity Bias of Multimodal-Aware Recommender Systems: A Modalities-Driven Analysis.” In: *MMIR@MM*. ACM, 2023, pp. 59–68.

[16] **Daniele Malitesta**, **Giuseppe Gassi**, Claudio Pomo, and Tommaso Di Noia. “Ducho: A Unified Framework for the Extraction of Multimodal Features in Recommendation.” In: *ACM Multimedia*. ACM, 2023, pp. 9668–9671.

[17] Yashar Deldjoo, Tommaso Di Noia, **Daniele Malitesta**, and Felice Antonio Merra. “A Study on the Relative Importance of Convolutional Neural Networks in Visually-Aware Recommender Systems.” In: *CVPR Workshops*. Computer Vision Foundation / IEEE, 2021, pp. 3961–3967.

[18] Yashar Deldjoo, Tommaso Di Noia, **Daniele Malitesta**, and Felice Antonio Merra. “Leveraging Content-Style Item Representation for Visual Recommendation.” In: *ECIR (2)*. Vol. 13186. Lecture Notes in Computer Science. Springer, 2022, pp. 84–92.

[19] **Felice Antonio Merra**, Vito Walter Anelli, Tommaso Di Noia, Daniele Malitesta, and Alberto Carlo Maria Mancino. “Denoise to Protect: A Method to Robustify Visual Recommenders from Adversaries.” In: *SIGIR*. ACM, 2023, pp. 1924–1928.

[20] Dario Di Palma, Vito Walter Anelli, **Daniele Malitesta**, Vincenzo Paparella, **Claudio Pomo**, Yashar Deldjoo, and Tommaso Di Noia. “Examining Fairness in Graph-Based Collaborative Filtering: A Consumer and Producer Perspective.” In: *IIR*. Vol. 3448. CEUR Workshop Proceedings. CEUR-WS.org, 2023, pp. 79–84.

[21] **Vito Walter Anelli**, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, **Claudio Pomo**, Francesco M. Donini, Eugenio Di Sciascio, and Tommaso Di Noia. “The Challenging Reproducibility Task in Recommender Systems Research between Traditional and Deep Learning Models.” In: *SEBD*. Vol. 3194. CEUR Workshop Proceedings. CEUR-WS.org, 2022, pp. 514–521.

[22] **Vito Walter Anelli**, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, **Claudio Pomo**, Francesco Maria Donini, and Tommaso Di Noia. “Elliot: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation.” In: *SIGIR*. ACM, 2021, pp. 2405–2414.

[23] **Vito Walter Anelli**, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, **Claudio Pomo**, Francesco Maria Donini, Eugenio Di Sciascio, and Tommaso Di Noia. “How to Perform Reproducible Experiments in the ELLIOT Recommendation Framework: Data Processing, Model Selection, and Performance Evaluation.” In: *IIR*. Vol. 2947. CEUR Workshop Proceedings. CEUR-WS.org, 2021.

[24] **Vito Walter Anelli**, **Daniele Malitesta**, **Claudio Pomo**, Alejandro Bellogín, Eugenio Di Sciascio, and Tommaso Di Noia. "Challenging the Myth of Graph Collaborative Filtering: a Reasoned and Reproducibility-driven Analysis." In: *RecSys*. ACM, 2023, pp. 350–361.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

With the advent of the Internet and the World Wide Web, several popular companies decided to invest into online platforms to advertise their products and/or services to customers all over the world. Most of people in today's society devotes an increasing portion of the daily routine into navigating through web pages offering, among the others, fashion items or electronic devices on e-commerce platforms, micro-videos, TV series, or movies on video streaming platforms, songs or podcasts on music streaming platforms, and restaurants or locations of interest on booking and tourism platforms.

Due to the ever-growing plethora of products/services offered on online platforms, customers might easily feel lost when surfing such heterogeneous and large catalogues, also considering that customers are not usually completely sure of their own preferences and tastes. Recommender systems (RSs) [260] have been efficiently employed to mitigate the so-called *information overload* problem, given their capability to profile users and items and filter only those products/services which might be of interest to customers. In this respect, the scientific literature enumerates diverse strategies and paradigms to address the recommendation task, from shallower solutions based upon heuristics and similarity measures [246, 259, 271] to more nuanced approaches which make use of the most recent advances in machine and deep learning to train powerful representation algorithms [126, 128, 160, 216, 371].

Among the most successful stories in recommendation, *collaborative filtering* (CF) [94] has settled as one of the most popular one, thanks to its easy rationales and applicability. Indeed, the core idea behind CF is that users which have interacted with the same items in the past might likely share tastes and preferences, thus justifying the recommendation of similar items in the near future. Particularly, the current literature regarding CF approaches for recommendation indicates that factorization-based models [126, 128, 160] represent the state-of-the-art solutions. In a nutshell, factorization-based approaches in CF work by mapping users and items in the system to embedded vectors in the latent space, and optimize a loss function driven by specific task or objective to pursue [216].

Despite the recognized success of CF techniques, recommendation still remains a challenging task. Among the open research challenges in the community, two of the most fundamental ones regard (i) the inexplicable nature of users' feedback [60], especially when it comes in implicit form (e.g., like/dislike, view/not-view); (ii) the useful and meaningful exploitation of the collaborative signal [325] in novel recommendation systems. In order to address (i) and (ii), diverse solutions have been proposed so far.

As for the (i) open challenge, there exist recommendation scenarios and domains, such as fashion [65, 361, 380], music [70, 236, 312], video [45, 67, 339], food [171, 222, 321], and tourism [274] recommendation, where the *multi-faceted* content characterizing items and users' preferences towards items can help generating more accurate recommendations with respect to traditional recommendation models which make use of the only information conveyed by the user-item interaction matrix. Recommendation algorithms leveraging *multimedia* content [189, 269, 309, 404] suitably embody the exploitation of *multimodal* side information [125, 338, 339, 382, 406, 407] (e.g., product images and descriptions, users' reviews, audio tracks) to enhance the representational power of embedded users' and items' profiles. By injecting high-level multimodal features extracted through deep learning models pre-trained for image classification [122], text sentiment analysis [256], and audio classification [130] into the recommendation downstream task, such models have continued to raise the performance limits of recommender systems for some time now.

Collaborative filtering (CF) promotes the idea of similar users interacting with similar items. Put into other words, the CF rationale may be defined as the *collaborative signal* of users and items interacting at *multiple* distance hops. Despite CF increasingly-popular application to the task of recommendation, the question is "are we really using the collaborative signal in the proper and most meaningful way?" [325]. When it comes to factorization-based approaches such as matrix factorization with Bayesian personalized ranking (MFBPR) [258], an argument might be said that the collaborative signal is only exploited in the formulation of the loss function (i.e., BPR). However, this is never employed to learn more refined users' and items' latent embeddings. In this respect, graph neural networks [51, 119, 410] are powerful and recent machine learning architectures which work on graph-like data (such as users and items interactions in a recommendation system) through the message-passing paradigm. That is, node representations (users and items) are iteratively refined through the representations of nodes from the *neighborhood*, at increasing distance hops. Graph-based recommender systems [32, 46, 126, 217, 248, 307, 324, 325, 343] have recently taken over personalized recommendation, provide incredibly-higher accuracy performance to traditional approaches in CF.

Multimodal-aware and graph-based recommendation systems are indubitably among the trending algorithms in recommender systems in the last few years. Thus, it becomes important, if not imperative, to consider how and to what extent integrating their diverse but complementary strategies into a unique recommendation framework. The idea is to identify the current pitfalls in both families of recommender systems, and combine them to help strengthening one another.

1.1 Thesis Statement

The content of this thesis is organized into thematic chapters, where each chapter reports on the background notions, analyses, and proposals from the papers which refer to that specific theme. Note that the papers ordering across the various chapters do not necessarily follow the chronological one; indeed, the idea was to provide an as much comprehensive and cohesive narrative as possible across the whole thesis, from multimodal-aware recommender systems and graph-based recommendation, to conclusively combine the two solutions.

To begin with, Chapter 2 and Chapter 3 provide the useful background notions and definitions regarding recommender systems and graph neural networks, respectively. Indeed, they represent the two main research topics of this thesis, and their formal presentation comes as essential to the other chapters in the thesis.

Then, the main research contributions are extensively described in Chapter 4, Chapter 5, Chapter 6, Chapter 7. Concretely, Chapter 4 and Chapter 5 are devoted to the formalization, theoretical/empirical analysis, and proposal of novel approaches in multimodal-aware recommendation; in a complementary manner, Chapter 6 provides a comprehensive introduction to graph-based recommendation, by considering different and novel evaluation dimensions which help unveiling unexpected and interesting aspects in the same topic; conclusively, Chapter 7 proposes a cohesive combination of multimodal-aware and graph-based recommender systems, through an insightful analysis of their multi-sided performance, and a novel graph-based recommendation approach which aims to overcome known issue in the related literature through the injection of multimodal content.

In conclusion, Chapter 8 wraps up the main take-home messages of this thesis. Intentionally, a separate chapter (i.e., Chapter 9) has been devoted to introducing the future research directions of this thesis. Indeed, we decide to provide intuitions, formalizations, and preliminary experimental results of the possible new research paths that naturally derive from the findings proposed in this thesis.

1.2 Research Contributions

The current section aims to offer a synthetic but comprehensive overview of the research contributions from this thesis, as organized into thematic chapters. For each of them, we briefly summarize the content, report on the related publications, and give complete details about the role of the Ph.D. candidate, **Daniele Malitesta**, in such publications. Note that, in all the papers cited in this section, and as already indicated in the thesis preamble, Daniele Malitesta is the **corresponding author**.

1.2.1 Chapter 4: Formalizing multimedia recommendation

Contributions

While recommendation systems leveraging multimedia content have long established as successful and efficient approaches in the literature, their application of *multimodal* deep learning strategies remains not clearly defined, formalized, and empirically analyzed. To this end, this chapter provides one of the first formal re-definition of multimedia recommendation under the lens of multimodal deep learning. A careful study of the related literature helps recognizing recurrent patterns in adopting multimodal strategies for multimedia recommendation to design a formal and unified scheme, conceptually applicable to exisiting multimedia recommender systems.

Publications

The chapter covers the topics presented and explored in “Formalizing Multimedia Recommendation through Multimodal Deep Learning” [205], currently under review at ACM Transactions on Recommender Systems Journal (TORS).

Ph.D. candidate’s role

Daniele Malitesta is the corresponding author of the paper [205].

1.2.2 Chapter 5: Leveraging the visual modality in multimedia recommendation

Contributions

The chapter deals with one of the possible application of multimodal-aware recommendation, that of visually-aware recommender systems. This family of recommender systems leverage the visual modality to enhance the representation of items and users' preferences towards items. The chapter opens with the proposal of a framework for the extraction of multimodal features in recommendation (Ducho) which can be easily and seamlessly integrated with V-Elliot, our proposed framework for rigorous and reproducible evaluation of visually-aware recommender systems. The so-built visual-based recommendation pipeline paves the way to multiple empirical analyses on the performance of such models, especially in terms of the pre-trained deep learning network adopted to extract high-level visual features from product images. The presented findings are eventually applied to two scenarios and settings: (i) fashion recommendation, with the proposal of a novel visually-aware recommender system which is capable of disentangling users' preferences at the granularity of content-style representation of product images; (ii) the application of adversarial attacks to product images to drive the recommendation of niche products close to the one of popular products, along with the evaluation on the efficacy of defensive countermeasures.

Publications

The chapter covers the topics presented and explored in "Ducho: A Unified Framework for the Extraction of Multimodal Features in Recommendation" [208], presented at the 31st International Conference on Multimedia (MM 2023); "V-Elliot: Design, Evaluate and Tune Visual Recommender Systems" [12], presented at the 15th Conference on Recommender Systems (RecSys 2021); "A Study on the Relative Importance of Convolutional Neural Networks in Visually-Aware Recommender Systems" [82], presented at the 4th Workshop on Computer Vision for Fashion, Arts, and Design, co-located with the 2021 Conference on Computer Vision and Pattern Recognition (CVPR 2021); "Leveraging Content-Style Item Representation for Visual Recommendation" [83], presented at the 44th European Conference on Information Retrieval (ECIR 2022); "Assessing Perceptual and Recommendation Mutation of Adversarially-Poisoned Visual Recommenders" [21], presented at the Discussion Papers co-located with the 20th International Conference of the Italian Association for Artificial Intelligence (AIxIA 2021).

Ph.D. candidate's role

Daniele Malitesta is the corresponding author of the papers [12, 21, 82, 83, 208].

1.2.3 Chapter 6: Evaluation of graph-based recommender systems

Contributions

The chapter delves into an extensive evaluation of graph-based recommendation. First, an out-of-the-box application extending the Elliot framework and using CUDA technologies with Docker is presented. Then, the framework is exploited to run a complete reproducibility study on selected graph-based recommender systems, indicating how traditional recommendation approaches, such as neighborhood-based ones, may unexpectedly outperform the selected graph-based solutions. Indeed, an analysis on the information conveyed by node degree at multiple distance hops unveils that the performance of (graph-based) recommender systems may depend on the properties of the dataset such models are trained on. These findings open to more careful and novel investigations on the possible dependences between the topological characteristics of recommendation datasets (measured as node degree, clustering coefficient, and degree assortativity) and recommendation performance of graph-based recommender systems. Conclusively, the chapter outlines a formal taxonomy of graph-based recommendation approaches from the literature, which recognizes node representation and neighborhood exploration as the main steps in the usual graph-based recommendation pipeline. The exploration of different solutions for the node representation and neighborhood exploration shows significant performance variations when considering accuracy and beyond-accuracy recommendation measures, separately and jointly in a multi-objective evaluation setting.

Publications

The chapter covers the topics presented and explored in "An Out-of-the-Box Application for Reproducible Graph Collaborative Filtering extending the Elliot Framework" [210], presented at the 31st Conference on User Modeling, Adaptation, and Personalization (UMAP 2023); "Challenging the Myth of Graph Collaborative Filtering: a Reasoned and Reproducibility-driven Analysis" [20], presented at the 17th Conference on Recommender Systems (RecSys 2023); "A Topology-aware Analysis of Graph Collaborative Fitering" [209], under review at the 2nd Learning on Graphs Conference (LoG 2023);

"How Neighborhood Exploration influences Novelty and Diversity in Graph Collaborative Filtering" [18], presented at the 2nd Workshop on Multi-Objective Recommender Systems, co-located with the 16th Conference on Recommender Systems (RecSys 2022); "Auditing Consumer- and Producer-Fairness in Graph Collaborative Filtering" [17], presented at the 45th European Conference on Information Retrieval (ECIR 2023); "Examining Fairness in Graph-Based Collaborative Filtering: A Consumer and Producer Perspective" [240], presented at the 13th Italian Information Retrieval Workshop (IIR 2023).

Ph.D. candidate's role

Daniele Malitesta is the corresponding author of the papers [17, 18, 20, 209, 210, 240].

1.2.4 Chapter 7: Graph-based recommendation exploiting multimodal information

Contributions

The ultimate objective of this thesis is to bring multimodality to graph-based recommendation to empower the latter with the high-level multimodal features enhancing the representation of users and items; the purpose is to tackle both conceptual issues (inexplicable nature of implicit feedback) and, consequently, algorithmical issues (such as oversmoothing). In this respect, this chapter provides an innovative analysis on the performance of state-of-the-art graph-based recommender systems leveraging multimodal information under several evaluation perspectives encompassing, among the others, novelty, diversity, bias, and fairness recommendation measures. On such a basis, the chapter ends with the proposal of a novel graph-based recommendation approach which leverages users' generated reviews as the textual features to characterize edges in the user-item bipartite graph. By revisiting the implicit feedback issue as a node representation error in the application of the message passing, the proposed solution adopts the sentiment conveyed by the reviews as a way to address the issue and, as positive side effect, mitigate the negative effects of oversmoothing.

Publications

The chapter covers the topics presented and explored in "Disentangling the Performance Puzzle of Multimodal-aware Recommender Systems" [206], presented at the 2nd Workshop on A Well-Rounded Evaluation of Recommender Systems (EvalRS), co-located with the 29th SIGKDD on Knowledge Discovery and Data Mining (KDD 2023); "On Popularity Bias of Multimodal-aware Recommender Systems: a Modalities-driven Analysis" [207], presented at the 1st Workshop on Deep Multimodal Learning for Information Retrieval (MMIR), co-located with the 31st International Conference on Multimedia (MM 2023); "Reshaping Graph Recommendation with Edge Graph Collaborative Filtering and Customer Reviews" [19], presented at the Workshop on Deep Learning for Search and Recommendation (DL2SR), co-located with the 31st ACM International Conference on Information & Knowledge Management (CIKM 2022).

Ph.D. candidate's role

Daniele Malitesta is the corresponding author of the papers [19, 206, 207].

1.3 Bibliographical Notes

For the sake of completeness, in this last section, we report on the publications which have been listed in the Publications section in the preamble of this thesis, but have not been cited among the research contributions. Specifically, we categorize them into two groups, namely: (i) research contributions regarding the main topics of this thesis that cannot properly be considered as paper publications (e.g., workshops and tutorials); (ii) other publications whose topics are similar/related to those of this thesis and where, *in same cases*, the Ph.D. candidate (Daniele Malitesta) is not the corresponding author.

Workshops and tutorials

- "First International Workshop on Graph-Based Approaches in Information Retrieval (IRonGraphs 2024)", accepted as workshop at the 46th European Conference on Information Retrieval (ECIR 2024).
- "Graph Neural Networks for Recommendation: Reproducibility, Graph Topology, and Node Representation" [211], accepted as tutorial at the 2nd Learning on Graphs Conference (LoG 2023).

Other publications

GNNs and KGs for recommendation

- "KGTORe: Tailored Recommendations through Knowledge-aware GNN Models" [212], accepted at the 17th Conference on Recommender Systems (RecSys 2023).

Adversarial attacks and defenses for visually-aware recommender systems

- "A Study of Defensive Methods to Protect Visual Recommendation Against Adversarial Manipulation of Images" [16], accepted at the 44th International SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021).
- "Adversarial Attacks against Visual Recommendation: an Investigation on the Influence of Items' Popularity" [22], accepted at the Workshop on Online Misinformation- and Harm-Aware Recommender Systems (OHARS), co-located with the 15th Conference on Recommender Systems (RecSys 2021).
- "Denoise to Protect: A Method to Robustify Visual Recommenders from Adversaries" [220], accepted at the 46th International SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2023).

Elliot, a framework for reproducibile recommender systems and their evaluation

- "The Challenging Reproducibility Task in Recommender Systems Research between Traditional and Deep Learning Models" [10], accepted at the 30th Symposium on Advanced Database System (SEBD 2022).
- "Elliot: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation" [11], accepted at the 44th International SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021).
- "How to Perform Reproducible Experiments in the ELLIOT Recommendation Framework: Data Processing, Model Selection, and Performance" [13], accepted at the 11th Italian Information Retrieval Workshop (IIR 2021). Evaluation

Chapter 2

Background on recommender systems

In the era of online platforms for e-commerce, media content delivery, tourism, food, and social networks, recommender systems play the key role of bridging the gap between customers' needs and products/services offered on such platforms. Specifically, recommender systems are (machine learning) algorithms aimed to exploit recorded user-item interactions to unveil hidden preference patterns, thus suggesting novel items to users. This chapter is devoted to the presentation and formalization of the personalized recommendation scenario. First, we provide the formal definitions for the recommendation task(s). Then, we deep dive into a widely-recognized taxonomy of popular recommendation approaches. Finally, we take into account the common recommendation pipeline, by focusing on three of its main stages. In this respect, we consider (i) the recommendation input to inject in the system, (ii) the optimization technique (in the case of model-based recommender systems) with an optional stage for negative sampling of items, and (iii) the evaluation protocol to evaluate the performance of any recommender system. The content of this chapter is inspired by [260] and other papers we will cite in the following.

2.1 Preliminaries

Let $\mathcal{U}$ and $\mathcal{I}$ be the sets of users and items in the recommendation system, respectively, where $|\mathcal{U}| = N$ and $|\mathcal{I}| = M$. Then, let $\mathbf{R} \in \mathbb{R}^{N \times M}$ be the user-item preference matrix, whose entries are continuous values such as the range $\{1, \ldots, 5\}$ (*explicit* feedback) or binary values such as $\{0, 1\}$ (*implicit* feedback). In the case of implicit feedback, note that the value 0 may not necessarily indicate that the user *disliked* a specific item; in a more broader sense, the value 0 stands for an item that has *not been interacted* by the user yet. In this respect, we may also define the set of items the user $u \in \mathcal{U}$ has interacted with (i.e., her positive items) as $\mathcal{I}_u^+ = \{i \in \mathcal{I} \mid R_{ui} = 1\}$. Trivially, the set of items the user has not interacted with yet is defined as $\mathcal{I}_u^- = \{i \in \mathcal{I} \mid R_{ui} = 0\} = \mathcal{I} \setminus \mathcal{I}_u^+$. Since the set of negative items $\mathcal{I}_u^-$ might be quite large for most of the users, we devote Section 2.3.2 to the presentation of the most popular techniques in the literature to perform negative sampling.

Based upon the notions and background concepts provided above, in the following, we formalize two of the most commont tasks in recommendation, namely, *rating prediction* and *top-k recommendation*. The two tasks are inherently and conceptually related, but show some differences we seek to outline.

2.1.1 Rating prediction

Let $u \in \mathcal{U}$ and $i \in \mathcal{I}_u^-$ be a user and one of her negative items, respectively. The task of rating prediction is about finding the function which maps the selected user and item to the predicted rating the user may express about the item. The function definition is:

$$\text{RATING PREDICTION: } \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}. \tag{2.1}$$

For example, the predicted rating for user u and her negative item i is obtained as:

$$\hat{R}_{ui} = \rho(u, i), \tag{2.2}$$

where $\rho(\cdot)$ is the rating prediction function defined above. Note that the task of rating prediction may be casted to a regression or binary classification task. Indeed, in an explicit feedback scenario, the regression task is the most suitable one to model rating prediction. On the contrary, in an implicit feedback setting, binary classification is the most suitable one to model rating prediction.

2.1.2 Top-*k* recommendation

Let $u \in \mathcal{U}$ be a user in the recommendation system, and k be a threshold to filter out items from the negative item setting of u, namely, $\mathcal{I}_u^-[k]$. The top-k recommendation task is defined as finding the utility function according to which the negative items of each user are sorted in descending order and filtered out to select the top-k negative items. The function definition is:

$$\text{TOP-k RECOMMENDATION: } \mathcal{U} \times \mathcal{I} \to \mathbb{R}. \tag{2.3}$$

For example, the predicted utility for user u and her negative item i is obtained as:

$$\hat{S}_{ui} = \sigma(u,i), \tag{2.4}$$

where $\sigma(\cdot)$ is the utility prediction function defined above. On such a basis, we retrieve, for each user, the list of k negative with the highest utility score:

$$u \to \begin{cases} i_1, \\ i_1, \\ \dots, \\ i_k \end{cases}, \tag{2.5}$$

where items have been re-indexed according to the values of $\hat{S}_{u*}$, such that $\hat{S}_{u0} > \hat{S}_{u1} > \dots > \hat{S}_{uk}$. Trivially, it becomes evident how the utility prediction function $\sigma(\cdot)$ and the rating prediction function $\rho(\cdot)$ may overlap. Indeed, The top-k recommendation task can be considered as one possible following step after the rating prediction task, where items are re-ordered for each user according to their predicted score.

It is also important to mention that most of the machine learning based approaches in recommendation which are designed and evaluated to pursue the task of top-k recommendation do not explicitly optimize a ranking-driven loss function during their training. Indeed, the common practice is to optimize a loss function which minimizes the error in predicting the interaction scores (i.e., rating prediction task), or an hybrid loss function which considers predicted scores for both positive and negative items. This aspect will be widely presented in Section 2.3.2.

2.2 Taxonomy of recommender systems

The history of recommender systems have known several approaches raising and settling as state-of-the-art solutions in the community. Despite the large plethora of recommendation techniques proposed so far, the literature recognizes three main families of recommender systems, that may be categorized according to the following taxonomy: (i) collaborative filtering approaches, (ii) content-based approaches, and

(iii) hybrid approaches. In the following, we present each of these recommendation families by describing their main ideas and formulations, along with some of the most popular techniques from the state-of-the-art.

2.2.1 Collaborative filtering approaches

Collaborative filtering [94] (CF) is currently among the prominent paradigms in recommendation. Recommendation approaches following the CF solution promote the idea that similar users (in terms of similar user profiles) may interact with similar items in the future. On such a basis, CF approaches heavily rely on the availability of user interactions with items. In the following, we present two of the main subcategories of recommender systems adopting the CF rationale, namely, neighborhood-based and model-based approaches.

Neighborhood-based approaches

Among the pioneer approaches in CF, neighborhood-based ones leverage the concept of users' or items' similarities based upon some heuristics.

For instance, ItemkNN [271] predicts whether a user u and an item i could interact depending on the similarity between i and the other items u has interacted with:

$$\rho^{\text{ItemkNN}}(u,i) = \sum_{j \in \mathcal{I}_u^+} sim(i,j), \tag{2.6}$$

where $\mathcal{I}_u^+$ is the set of items interacted by user u, while $sim(\cdot)$ is a similarity function (e.g., the cosine similarity) computed between the i-th and j-th column vectors of the user-item interaction matrix. For computational purposes, often times the set $\mathcal{I}_u^+$ is restricted to the k-most similar items.

The dual approach, UserkNN [259], estimates the interaction score of (u,i) based upon the similarity between u and the other users i has been interacted by:

$$\rho^{\text{UserkNN}}(u,i) = \sum_{v \in \mathcal{U}_i^+} sim(u,v), \tag{2.7}$$

where, complementarily to above, $\mathcal{U}_i^+$ is the set of users that interacted with item i.

Finally, another famous and efficient solution in this family of recommender systems is RP$^3\beta$ [246]. This technique is based upon the random walk process performed within the user-item interaction graph. Specifically, the authors discuss about the possibility to model the user navigation throughout the item catalogue through a 3-hops random walk. That is, user u starts by exploring one of the interacted items at random (1st hop), then goes through one of the users who interacted with that itemm (2nd hop), and finally picks one of the other items that user has interacted with (3rd hop). Let $\mathbf{D} \in \mathbb{R}^{(N+M)\times(N+M)}$ be the diagonal matrix where each element in the diagonal is the degree of a user or an item. The probability matrix $\mathbf{P} \in \mathbb{R}^{(N+M)\times(N+M)}$ of a user picking any of the interacted item is given by:

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}, \tag{2.8}$$

where $\mathbf{A} \in \mathbb{R}^{(N+M)\times(N+M)}$ is the adjacency matrix obtained from the user-item interaction matrix $\mathbf{R}$ (refer to Section 3.1.2 for a general description of the adjacency matrix for graph neural network models). By generalizing, after l random walk hops, the probability matrix is calculated as:

$$\mathbf{P}^h = (\mathbf{D}^{-1}\mathbf{A})^h. \tag{2.9}$$

On such a basis, the final formulation the authors provide is given by:

$$\rho^{\mathrm{RP}^3\beta}(u,i) = \frac{p^3_{ui}}{d^{\beta}_{ii}}, \tag{2.10}$$

where p^3_{ui} is the entry of the $\mathbf{P}^3$ matrix corresponding to user u and item i, while d_{ii} is the i-th entry of the diagonal matrix $\mathbf{D}$ with $\beta \in \mathbb{R}$, $\beta > 0.0$. Note that the β factor serves as flattening term to promote the *diversity* of recommended items (see Section 2.3.3). When calculating the utility score for user u and two items i and j, and assuming that $p^3_{ui} \simeq p^3_{uj}$ (i.e., the probability of reaching the two items through random walk is almost equal), the predicted utility score should not be penalized for that one item having a smaller degree (i.e., smaller number of recorded interactions).

Model-based approaches

With the advent of machine and deep learning, model-based recommender systems have increasingly taken over personalized recommendation. Differently from neighborhood-based approaches, such solutions exploit the user-item interaction matrix to learn a parametric rating/utility function to predict the rating/utility score of any user-item pair. Among the various techniques proposed in the literature, factorization-based ones have been largely adopted over the years. The main rationale is to represent users' and items' profiles through latent embeddings, and learn their weights by optimize some loss function (see Section 2.3.2).

Specifically, we indicate with $\mathbf{e}_u \in \mathbb{R}^d$ and $\mathbf{e}_i \in \mathbb{R}^d$ the embeddings for user u and item i and, generally, $d \ll N, M$. One of the widely-popular approaches, matrix factorization [160] (MF), estimates the user-item interaction score through the dot product of their embeddings:

$$\rho^{\mathrm{MF}}(u,i) = \mathbf{e}_u^\top \mathbf{e}_i. \tag{2.11}$$

By leveraging the representational power of deep neural networks, in neural collaborative filtering [128] (NCF), the authors propose to predict the user-item interaction score through the following:

$$\rho^{\mathrm{NCF}}(u,i) = \sigma(\mathbf{W} \cdot (mlp([\mathbf{e}_u, \mathbf{e}_i]) + \mathbf{e}_u \odot \mathbf{e}_i)), \tag{2.12}$$

where $\sigma(\cdot)$ is an activation function (e.g., the sigmoid), $\mathbf{W}$ is a weight matrix, $mlp(\cdot)$ is a multilayer perceptron where the input is the concatenation of the user and item embeddings, and $\odot$ is the element-wise product. In SimpleX [216], the authors introduce a score function which is based upon the importance of the items interacted by each user:

$$\rho^{\mathrm{SimpleX}}(u,i) = \sigma(\alpha \mathbf{e}_u + (1-\alpha)\mathbf{W} \cdot \mathbf{e}'_u), \tag{2.13}$$

where $sim(\cdot)$ is a similarity function (e.g., the cosine similarity), α is a weighting hyper-parameter, $\mathbf{W}$ is a weight matrix, and $\mathbf{e}'_u$ is obtained as the weighted aggregation (e.g., through attention) of the embeddings of the items interacted by u.

2.2.2 Content-based approaches

Differently from collaborative filtering approaches in recommendation, *content-based* recommender systems [233] leverage the content information describing users and items in the system. In the case of items, they are generally represented through attribute metadata, or through any other source of structured data such as knowledge graphs or images/texts. Recommendation in content-based recommender systems usuall involves three core steps [38, 107], namely: (i) representation of items' content in a proper and meaningful description space; (ii) learning of users' profile by collecting their preference data in the form of interacted items' content and trying to generalize on it; (iii) items' filtering to suggest relevant items to each user according to her preference. Based upon the way such algorithms are designed, they can effectively be used to tackle some issues in the literature such as the cold-start issue (see later), as each (new) item in the system comes with its descriptive content. At the same time, learning users' profiles through the items' content may drive the algorithm towards an *overspecialization* scenario where the recommender tends to recommend items which are too much homogeneous to the ones the user has already interacted with; this may undermine other objectives such as *diversity* and *novelty* of recommendations (see Section 2.3.3).

2.2.3 Hybrid approaches

As their name suggest, *hybrid* approaches [7] in recommendation are designed to incorporate the positive aspects of both collaborative filtering and content-based recommendation systems. Indeed, most of the recent and current solutions in recommendation, especially when it comes to multimedia- and knowledge-aware recommender systems, may be defined as hybrid approaches. In this respect, all the recommender systems which are going to be presented in this dissertation, both the evaluated ones and the proposed ones, should be intended as hybrid recommender systems.

2.3 The recommendation pipeline

The usual recommendation pipeline to find the suitable rating/utility function may depend on the specific recommendation approach. In the case of model-based approaches (using machine learning techniques), the pipeline essentially comprises three stages [11], namely: (i) the input processing and injection into the recommender system, (ii) the model optimization through a loss function and the optional items' negative sampling, and (iii) the evaluation of the performance. On the contrary, as for neighborhood-based approaches, only (i) and (iii) take place. In the following, we explore and describe each of these stages of the pipeline.

2.3.1 Recommendation input

Providing meaningful input data to the recommendation system is of the utmost importance to produce high-quality recommendations. Depending on the type of recommender system, the injected inputs may change. Nevertheless, in the simplest configuration, any recommender system takes as input the user-item interaction matrix $\mathbf{R}$, as well as the list of users and items in the system.

Then, when considering content-based or hybrid recommender systems, users and items may come with additional content or side information which provide further description of their profiles, independently on the recorded user-item interactions. For instance, in specific scenarios and domains such as fashion recommendation, items' representation can be enhanced through visual/textual features extracted from product images/descriptions. For other similar examples, consider again the cases presented in Section 2.2.2 and Section 2.2.3.

Finally, data input might undergo some pre-processing operations. While the existing literature enumerates a wide range of approaches for data pre-processing, here we describe one the most popularly-adopted one, namely, the p-core. Before presenting the p-core, it becomes essential to provide a formal distinction between *cold/warm* users and items. Under a certain threshold p, *warm* (*cold*) users are those users who have more (less) than p interactions; trivially, *warm* users are the most active ones on the platform, while *cold* users are the least active ones. On the item side, the definition applies similarly. That is, under a certain threshold p, *warm* (*cold*) items are those items having more (less) than p interactions in the systems; trivially, *warm* items are the most popular ones, *cold* items are the least popular ones. Note that other definitions of *warm/cold* users and items exist; however, the one presented above is strictly related to the formulation of the p-core we introduce in the following.

In this sense, the p-core technique can be applied to both users and items in the recommendation system. In general, the p-core strategy filters out users (items) which have less than p interactions in the user-item interaction matrix. This approach is usually adopted to retain only *warm* users (items) in the system, and it is strictly related to the known issue of cold-start. Indeed, when keeping only users having more than p interactions in the system, we allow the recommendation algorithm to uncover preference patterns from a richer source of data. Conversely, without the application of the p-core, all users are taken into account during the designing of the recommendation approach; indeed, the recommender system may be not capable to infer preference predictions over *cold* users as much as it does on *warm* users. Trivially, the dual is true on the item side.

2.3.2 Optimization and negative sampling

This pipeline stage [216] is involved only in the case of model-based recommendation approaches. As recognised in the related literature, approaches in recommendation mainly adopt five *training objectives*. Note that, in some of the described optimizations, strategies for items' negative sampling are adopted.

Indeed, the most popular one is Bayesian personalized ranking (BPR), whose rationale drives from the concept of interacted (i.e., positive) and non-interacted (i.e., negative) items for each user in the catalogue. Let $\mathcal{T} = \{(u,i,j) \mid i \in \mathcal{I}_u^+ \wedge j \in \mathcal{I}_u^-\}$ be a set of triples including, for each user u, a positive and a negative item. BPR seeks to maximize the posterior probability of each user u preferring a positive item i over a negative item j. Thus, the BPR loss function is calculated as:

$$\mathcal{L}^{\text{BPR}}(\Theta) = -\sum_{(u,i,j)\in\mathcal{T}} \ln\, \sigma(\hat{R}_{ui} - \hat{R}_{uj}), \tag{2.14}$$

where Θ is the vector of all model's weights (e.g., in the case of MF, the user and item embeddings), $\sigma(\cdot)$ is the sigmoid function, and $\hat{R}_{ui}, \hat{R}_{uj}$ are the predicted scores for the pairs of user u with the positive item i, and user u with the negative item j.

The pairwise hinge loss (PH) works by maximizing the distance between the user-positive item pair and the user-negative item pair, at least under a certain marginal threshold. Let $\mathcal{T}^+ = \{(u,i) \mid i \in \mathcal{I}_u^+\}$ and $\mathcal{T}^- = \{(u,j) \mid j \in \mathcal{I}_u^-\}$ be the sets of pairs of users and their positive items, and user and their negative items. Then, the PH loss is defined as:

$$\mathcal{L}^{\text{PH}}(\Theta) = \sum_{(u,i)\in\mathcal{T}^+}\sum_{(u,j)\in\mathcal{T}^-} w_{ui}[m + ||\mathbf{e}_u - \mathbf{e}_i||^2 - ||\mathbf{e}_u - \mathbf{e}_j||^2]_+, \tag{2.15}$$

where w_{ui} is a ranking loss weight, $m > 0$ is the margin size, and $[x]_+ = max(x,0)$.

The binary cross-entropy loss (BCE) is adopted for the task of binary classification, discerning whether an item should be classified as positive or negative for a given user:

$$\mathcal{L}^{\text{BCE}}(\Theta) = -\sum_{(u,k)\in\mathcal{T}^+\cup\mathcal{T}^-} R_{uk}\ln(\hat{R}_{uk}) + (1 - R_{uk})\ln(1 - \hat{R}_{uk}), \tag{2.16}$$

where $R_{uk} = 1$ if $k \in \mathcal{I}_u^+$, 0 otherwise. The multiclass version of the previous loss implies the recast of the item recommendation problem to a multiclass classification one where, for each user, the model predicts which of the items (from the whole catalogue) the user will likely interact with. Such a loss function is named softmax cross-entropy loss (SCE), and is calculated as:

$$\mathcal{L}^{\text{SCE}}(\Theta) = -\sum_{u\in\mathcal{U}}\sum_{k\in\mathcal{I}} R_{uk}\ln\left(\frac{\exp(\hat{R}_{uk})}{\sum_{v\in\mathcal{I}}\exp(\hat{R}_{uv})}\right), \tag{2.17}$$

where $\exp(\cdot)$ is the exponential function.

The mean square error (MSE) measures the distance between the true and predicted score for each user-item pair:

$$\mathcal{L}^{\mathrm{MSE}}(\Theta) = \sum_{(u,k)\in\mathcal{T}^+\cup\mathcal{T}^-} (R_{uk} - \hat{R}_{uk})^2. \tag{2.18}$$

While we defined $\mathcal{I}_u^-$ as the set of *all* negative items for user u, oftentimes it is computationally-expensive to work with such a large set of elements. For this reason, the common approach is to sample a subset of negative items for each user, namely, *negative sampling*. Indeed, the authors of BPR suggest to sample, for each pair of user-positive item, only one negative item for that user. Nevertheless, the solution can be trivially generalized to sampling M^- negative items for each user.

Let $\hat{\mathcal{I}}_u^-$ be the set of sampled negative items for user u, with $|\hat{\mathcal{I}}_u^-| = M^-$. An additional training objective which have been successfully adopted over the last few years in the literature is the contrastive loss. For instance, the authors in SimpleX propose a cosine contrastive loss (CC) defined as:

$$\mathcal{L}^{\mathrm{CC}}(\Theta) = \sum_{(u,i)\in\mathcal{T}^+} (1 - \hat{R}_{ui}) + \frac{w}{M^-} \sum_{j\in\hat{\mathcal{I}}_u^-} [\hat{R}_{uj} - m]_+, \tag{2.19}$$

where m is a margin size as in the PH loss, while w controls the relative weights of the two addendum of the loss.

2.3.3 Evaluation

The final stage of any recommendation pipeline is the evaluation of the performance of the recommender system. In the following, we analyze two major aspects, regarding the (i) dataset splitting paradigm, and (ii) the metrics formulation.

Dataset splitting

Before introducing the formulations for all evaluation metrics, it is fundamental (as in any machine learning task) to separate the data into three sets, defined as *train*, *validation*, and *test* sets. In this respect, the literature enumerates different strategies for dataset splitting.

As already done in previous sections of this chapter, we remind just few of them, according to their popularity. Generally, the main strategies for dataset splitting are twofold, depending on whether they are performed via a (i) *random* splitting or (ii) *time-aware* splitting.

In the *random* setting, user-item interactions are split in a random manner into train, validation, and test sets. This random splitting may be random on the *whole* set of user-item interactions, or may be performed at *user-level.* In this latter case, interactions are split so that, for each user, a certain percentage of her interactions stay in the train, validation, and test sets (e.g., 80%, 10%, and 10% as a general practice). Trivially, it becomes evident that in the first scenario, it might happen that not all initial users are retained in the training, validation, and test sets. On the contrary, when performing a random splitting on user-level, it is quite likely the whole set of initial users will be present in all the three sets.

Conversely, the *time-aware* splitting requires an indication of when any user-item interaction occurred. On such a basis, the initial user-item interaction data is chronologically sorted on user-level. This allows to split the dataset into a train set where, for each user, interactions occurred before a specific time threshold; in the validation and test sets, instead, the remaining interactions are collected. An argument might be made that this time-aware splitting is the one which better depicts a real-world scenario. Indeed, the recommendation system should be ideally able to predict novel user-item interactions in the future, starting from the ones performed in the past. Nevertheless, it should be noted that this splitting requires the time information to be recorded along with each user-item interaction; unfortunately, this is the case for several datasets in the current literature.

When splitting the original dataset into train, validation, and test, it might be the case that the users and items appearing in each of them may not be overlapping sets. Indeed, by indicating $\mathcal{U}_{train}$, $\mathcal{U}_{val}$, and $\mathcal{U}_{test}$ as the sets of users belonging to the train, validation, and test set, respectively, we have that $\mathcal{U}_{train} \cup \mathcal{U}_{val} \cup \mathcal{U}_{test} = \mathcal{U}$. However, in the most general setting, we may have that $\mathcal{U}_{train} \neq \mathcal{U}_{val} \neq \mathcal{U}_{test} \neq \mathcal{U}$ (note that this cannot happen when performing a random/temporal-aware splitting on user-level). On the item side, when following any of the dataset splitting introduced above, it is quite likely that not all item sets in the train, validation, and test are overlapped. That is, $\mathcal{I}_{train} \neq \mathcal{I}_{val} \neq \mathcal{I}_{test} \neq \mathcal{I}$. Since in the common settings used to train and evaluate recommender systems the learning is *transductive*, the recommendation algorithm cannot be tested over users and/or items which are not present in the train set. Thus, whatever the type of evaluation, the evaluation stage should first remove from the validation/test sets those interactions involving users and/or items which do not belong to the train set (the one which is known by the model in advance).

Recommendation metrics

The selection of the proper metric(s) to perform the model evaluation is highly dependent on the recommendation task to be pursued, namely, either rating prediction or top-k recommendation. Moreover, the recent literature has raised the urge to test the performance of recommender systems under metrics other than the commonly-adopted *accuracy* ones. Such class of metrics is better known as *beyond-accuracy* metrics, and capture other recommendation objectives. For instance, it might be interesting to assess the ability of the recommender system to generate *novel* [310, 311] and *diverse* [108, 276] recommendation lists for each user; moreover, in the recent need to tackle issues related to *bias* and seek *fairness* in recommendation [3, 26, 39, 81, 141], it could be useful to evaluate the performance of a recommender system under these perspectives. In the following, we provide a twofold categorization of the most commonly-adopted metrics, based upon: (i) the recommendation task, and (ii) accuracy/beyond-accuracy.
Accuracy metrics. When dealing with the task of rating prediction, any error-based metric can be helpful to evaluate the performance of a recommender system. Let $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ be the sets of user-item interactions in the train, and test sets, respectively. The rating prediction task may be evaluated through the mean absolute error (MAE), mean squared error (MSE), or the root mean squared error (RMSE) formulated as follows:

$$\text{MAE} = \frac{1}{|\mathcal{D}_{test}|} \sum_{(u,i)\in\mathcal{D}_{test}} abs(\hat{R}_{ui} - R_{ui}), \tag{2.20}$$

$$\text{MSE} = \frac{1}{|\mathcal{D}_{test}|} \sum_{(u,i)\in\mathcal{D}_{test}} (\hat{R}_{ui} - R_{ui})^2, \tag{2.21}$$

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{D}_{test}|} \sum_{(u,i)\in\mathcal{D}_{test}} (\hat{R}_{ui} - R_{ui})^2} = \sqrt{\text{MSE}}, \tag{2.22}$$

where $abs(\cdot)$ is the absolute value operation.

When dealing with the task of top-k recommendation, multiple possible evaluation paradigms exist. Among them, we remind the *all-unrated items* one. Specifically, under this paradigm, the first step is to predict the *whole* user-item interaction matrix through the recommendation algorithm. Second, a list of recommended items is generated for each user, based upon the predicted score for each item in the train set. As this list may be quite large, it is common practice to retain only the top-k items according to the predicted ratings. Finally, accuracy recommendation metrics are calculated, by considering the generated list of k items, and the items each user has interacted with in the test set. We define such sets as $\mathcal{I}_{test}^{(u)}$, while the list of k recommended items for user u is indicated as $\mathcal{I}_{rec}^{(u)}$. In the following, we report the formulation for the recall (Recall@k), hit ratio (HR@k), precision (Precision@k), and normalized discounted cumulative gain (nDCG@k):

$$\text{Recall@}k = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \frac{|\mathcal{I}_{test}^{(u)} \cap \mathcal{I}_{rec}^{(u)}|}{|\mathcal{I}_{test}^{(u)}|}, \tag{2.23}$$

$$\text{HR@}k = \frac{|\mathcal{U}_{hits@k}|}{|\mathcal{D}_{test}|}, \tag{2.24}$$

$$\text{Precision@}k = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \frac{|\mathcal{I}_{test}^{(u)} \cap \mathcal{I}_{rec}^{(u)}|}{|\mathcal{I}_{rec}^{(u)}|}, \tag{2.25}$$

$$\text{nDCG@}k = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}, \tag{2.26}$$

where $\mathcal{U}_{hits@k} = \{u \in \mathcal{U}_{test} \mid \mathcal{I}_{test}^{(u)} \cap \mathcal{I}_{rec}^{(u)} \neq \emptyset\}$, while $rel_i = 1$ if $i \in \mathcal{I}_{test}^{(u)}$, 0 otherwise.

For the sake of completeness, we also report two recommendation metrics which are used to measure the accuracy of recommendation, but when considering the whole set of recommendable items (not only up to the k-th item in the recommendation list). They are the area under curve (AUC) and the average recall (AR), formulated as follows:

$$\text{AUC} = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \frac{1}{|\mathcal{I}_{test}^{(u)} \times \mathcal{I} \setminus \mathcal{I}_{test}^{(u)}|} \sum_{(i,j) \in \mathcal{I}_{test}^{(u)} \times \mathcal{I} \setminus \mathcal{I}_{test}^{(u)}} \mathbb{1}(\hat{R}_{ui} > \hat{R}_{uj}), \tag{2.27}$$

$$\text{AR} = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \frac{1}{|\mathcal{I}_{test}^{(u)}|} \sum_{k=1}^{|\mathcal{I}_{test}^{(u)}|} rel(k), \tag{2.28}$$

where $\mathbb{1}(\cdot)$ is the function returning 1 if the input condition is true, 0 otherwise, while $rel(k)$ is 1 if the item in the k-th position is relevant for user u, 0 otherwise.

Beyond-accuracy metrics. In this section, we explore four families of beyond-accuracy recommendation metrics accounting for the *novelty* and *diversity* of the produced recommendation lists, along with those regarding the concepts of *bias* and *fairness* in recommendation.

In terms of recommendation novelty [311], we recall the expected popularity complement (EPC) and the expected free discovery (EFD). First, we define the sets of seen, relevant, and chosen items as $\mathcal{I}_{seen}^{(u)}$, $\mathcal{I}_{rel}^{(u)}$ and $\mathcal{I}_{chosen}^{(u)}$, where $\mathcal{I}_{chosen}^{(u)} = \mathcal{I}_{seen}^{(u)} \cap \mathcal{I}_{rel}^{(u)}$.

Then, the formulations for EPC@k and EFD@k are:

$$\text{EPC@}k = \frac{c}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \sum_{i_k \in \mathcal{I}_{rel}^{(u)}} p(seen \mid k, u, \mathcal{I}_{rel}^{(u)}) p(rel \mid i_k, u)(1 - p(seen \mid i_k)), \tag{2.29}$$

$$\text{EFD@}k = \frac{1}{|\mathcal{U}_{rel}|} \frac{1}{|\mathcal{I}_{rel}^{(u)}|} \sum_{i \in \mathcal{I}_{rel}^{(u)}} log_2 p(i \mid seen), \tag{2.30}$$

where $p(test)$ and $p(rel)$ stand for the probability of an item to be seen and relevant, respectively. Conceptually, the two measures indicate the expected number of recommended unknown items which are also relevant, and the expected number of recommended known items which are also relevant, respectively.

In terms of recommendation diversity [108, 276], we recall the gini index (Gini@k) and the Shannon Entropy (SE@k). For the Gini@k, we first define:

$$\text{Gini@}k_{\downarrow} = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \frac{1}{k-1} \frac{\sum_{i=1}^{|\mathcal{I}_{rec}^{(u)}|} (2i - |\mathcal{I}_{rec}^{(u)}| - 1) times(i)}{\sum_{i=1}^{|\mathcal{I}_{rec}^{(u)}|} times(i)}, \tag{2.31}$$

where $times(i)$ is a function returning the number of times item i appears in the recommendation lists. Gini@$k_{\downarrow}$ has values close to 0 when diversity of recommended items is high, meaning that the probability of all items being recommended is almost the same. Given that we usually adopt metrics formulations which adhere to the principle *higher is better*, we will use the version Gini@$k = 1 -$ Gini@$k_{\downarrow}$. As for the SE@k, we have:

$$\text{SE@}k = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} - \sum_{i=1}^{|\mathcal{I}_{rec}^{(u)}|} \frac{times(i)}{\sum_{i=1}^{|\mathcal{I}_{rec}^{(u)}|} times(i)} ln \left(\frac{times(i)}{\sum_{i=1}^{|\mathcal{I}_{rec}^{(u)}|} times(i)} \right), \tag{2.32}$$

where SE@k is 0 when a single item is always recommended, and it reaches the value $ln(|\mathcal{I}_{test}^{(u)}|)$ when all items are recommended with the same frequency.

In the intersection between recommendation novelty and diversity, we should remember the concept of *coverage*. Such a property can be measured through the item coverage (*iCov*@k), to be intended as the number of different items which are covered across all recommendation lists for all users in the system. Its formulations is given by:

$$\text{iCov@}k = \frac{|\bigcup_{u \in \mathcal{U}_{test}} \mathcal{I}_{rec}^{(u)}|}{\mathcal{I}_{test}}. \tag{2.33}$$

When it comes to bias in recommendation, several different definitions may be taken into account. One of the most common ones regards items' popularity bias, and it needs the preliminary definition of *short head* and *long tail* items, which is highly related to the warm/cold items definition we provided above. By plotting item IDs from the most popular (with highest number of interactions) to the least popular on the x axis, and the items' popularity on the y axis, one may note that popular items are much fewer than niche ones. To statistically split items into most and least popular ones, we may apply the 80%/20% Pareto principle, according to which the 80% or items' popularity is cumulatively provided by the 20% of the items in the catalogue (i.e., the popular ones); similarly, the 20% of popularity is cumulatively provided by the 80% of items in the catalogue (i.e., the niche ones). From a visual inspection of the 2D plot mentioned above, popular items belong to the so-called *short head* ($\mathcal{I}_{sh}$). On the contrary, the niche items belong to the *long tail* ($\mathcal{I}_{lt}$). Based upon the concepts of *short head* and *long tail* items, in the following, we recall three commonly-adopted recommendation metrics to measure the ability of a recommender system to retrieve items from the *long tail.* Indeed, one of the purposes of any *provider* is to sell as much numerous as possible items from the catalogue, especially the niche ones, since the popular items are the ones which have much higher chances to be recommended and sold to the users. That is also why items' popularity bias is sometimes referred to as *provider fairness.* The three metrics [3] we present are the average recommendation popularity (ARP@k), the average percentage of long tail items (APLT@k), and the average coverage of long tail items (ACLT@k):

$$\text{ARP@}k = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \frac{\sum_{i \in \mathcal{I}_{rec}^{(u)}} pop(i)}{|\mathcal{I}_{rec}^{(u)}|}, \tag{2.34}$$

$$\text{APLT@}k = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \frac{\{|i \mid i \in \mathcal{I}_{lt} \cap \mathcal{I}_{rec}^{(u)}|\}}{|\mathcal{I}_{rec}^{(u)}|}, \tag{2.35}$$

$$\text{ACLT@}k = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \sum_{i \in \mathcal{I}_{rec}^{(u)}} \mathbb{1}(i \in \mathcal{I}_{lt}), \tag{2.36}$$

where $pop(\cdot)$ is the function returning the popularity (number of interactions) of an item, while $\mathbb{1}(\cdot)$ is the function returning 1 if the item belongs to the long tail, 0 otherwise. First, the ARP@k represent the average popularity of the recommende items: the lower the better, as this means more items from the long tail are retrieved. Then, the APLT@k stands for the average percentage of items from the long tail: the higher the better, meaning that more recommended items are from the long tail.

Finally, the ACLT@k is the average number of recommended items from the long tail: again, the higher the better.

To conclude this chapter, we present the recommendation metrics accounting for *fairness* in recommendation. As already stated for bias, recommendation fairness may refer to several concepts and meanings. For the sake of this thesis and chapter, we are now referring to the concept of producing recommendations which are fair to all users groups in the system. Similarly to the bias setting, also in this case it becomes fundamental to formally define a criterion to split users into groups. Once again, the 80%/20% Pareto principle comes in handy, as it works similarly to the previous case, but with a different meaning. Indeed, in the case of users, we are categorizing users into those who are highly active on the platform (warm users) and those who are least active on the platform (cold users). The careful reader may have noticed that we already gave such a definition in the previous sections of this chapter, but adopting another mathematical criterion (i.e., the p-core strategy). Given that this definition of fairness accounts for users, it is oftentimes referred to as *consumer fairness*. In this respect, the idea is that users on a platform should receive recommendations with equal quality independently on their activity level on the platform. To measure this property on recommender systems [81], one possibility is to adopt the mean absolute deviation over users for ratings (UMADrat@k), whose formulation we report in the following:

$$\text{UMADrat@}k = \underset{u \in \mathcal{U}_{cold}, v \in \mathcal{U}_{warm}}{avg} (\text{MAD}(\hat{R}_{u*}, \hat{R}_{v*})), \tag{2.37}$$

where $avg(\cdot)$ is the function taking the average over all users from the cold set $\mathcal{U}_{cold}$ and warm set $\mathcal{U}_{warm}$, while $\text{MAD}(\hat{R}_{u*}, \hat{R}_{v*})$ is the mean absolute deviation between the average rating received by cold and warm users, respectively. Note that the formula may be trivially extended to rankings instead of ratings.

Chapter 3

Background on graph neural networks

Graph neural networks have revolutionized machine and deep learning over the last decade. As the name suggests, such neural architectures are especially tailored to work on graph-structured data, such as social networks, citation networks, knowledge graphs, molecules, and recommendation networks. As a useful background to understand graph-based recommendation approaches (one of the core topics in this thesis), we devote this chapter to providing a brief overview of the main definitions and concepts behind graph neural networks. In this respect, we first introduce some notions about graphs as data structures. Second, we introduce the message passing iterative algorithm, which represents the most atomic building block of any graph neural network model. Then, we recall some of the most popular graph neural network architectures in the recent literature. Finally, we describe and formalize the main tasks graph neural networks are currently trained on. The content of this chapter is inspired by the following papers and books in the literature [51, 119, 410].

3.1 Basic notions about graphs

This first section of the chapter addresses some basic notions about graphs as data structures. After a formal definition of what a graph is, along with its multiple variations, we formalize the adjacency matrix and node features, which constitute the main components to define a graph.

3.1.1 Definition of graph

We define a graph through the set of nodes $\mathcal{V}$ and edges $\mathcal{E}$ connecting such nodes, namely, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Specifically, an edge between node $v \in \mathcal{V}$ and $w \in \mathcal{V}$ exists if the two nodes are connected. When edges in a graph come with a direction (e.g., node v is connected to node w but not viceversa), the graph is said to be *directed*; otherwise, if the edge direction is not defined (e.g., v is connected to w and also the viceversa holds true) the graph is said to be *undirected*. Nodes in a graph may belong to disjoint partitions, namely, $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2, \dots, \cup \mathcal{V}_p$, having $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset, \forall i \neq j$. This type of graph, named *heterogeneous*, has edges which may connect nodes from different partitions following a specific rationale. However, when edges only connect nodes from different partitions, the graph is said to be *multipartite*. A special case of such graphs is the *bipartite* one, where nodes belong to two different partitions, and nodes from one partition can only be connected to nodes from the other partition. A classical example of bipartite graph is the user-item recommendation graph on e-commerce platforms, where users may only interact with items and viceversa (the graph is also undirected).

3.1.2 Adjacency matrix

An adjacency matrix represents the graph $\mathcal{G}$ into matrix format. By assigning each node in the graph an ordering (i.e., nodes are represented by specific rows and columns in the matrix), the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is designed such that $A_{v,w} = a$ if there exists an edge connecting v and w, 0 otherwise. In a directed graph, $A_{vw} \neq A_{wv}$ in general, while in an undirected graph, $\mathbf{A}$ is symmetric and $A_{vw} = A_{wv}$. Note that the entry $A_{vw} = a$ is a real-valued number which represents the weight of the edge connecting the nodes v and w (for instance, the value may be positive or negative when the graph is directed to indicate the different edge direction). For the sake of simplicity, and in general, $A_{vw} = \{0, 1\}$, where the edge weight is discretized to represent the absence or presence of an edge, respectively. Finally, note also that in multipartite graphs (e.g., bipartite graphs) there exist entire portions of the adjacency matrix whose entries are zero, as in such a family of graphs some node partitions are not connected one another. It is also important to introduce the concept of *neighborhood* of a node v, defined as the set of nodes which are connected to v through one edge. Let $\mathcal{N}(v) = \{w \in \mathcal{V} \mid A_{vw} = 1\}$ be the neighborhood of v. Then, we calculate the *degree* of v as $|\mathcal{N}(v)|$ or, alternatively, $\sum_{w \in \mathcal{N}(v)} A_{vw}$.

In the training of graph neural networks (see later), it is common practice to consider the symmetric normalized adjacency matrix (i.e., $\mathbf{A}_{sym}$). Starting from the definition of adjacency matrix, we calculate $\mathbf{A}_{sym}$ as:

$$\mathbf{A}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}. \tag{3.1}$$

The purpose of this normalization is to flatten the weight importance of all nodes in the graph based upon their degree to tackle possible instabilities during the optimization of the graph neural network.

3.1.3 Node features

Nodes in a graph may be associated with attributes or features describing them. Generally, such features are formally represented through a real-valued matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ where d is the dimensionality of the feature matrix. While $\mathbf{X}$ may consist of features describing real-world characteristics of the nodes (e.g., continuous and categorical features representing patients in a patients-diseases bipartite graph), it may also stand for embeddings in the latent space which do not have a direct connection to real-world properties (e.g., vector embeddings such as in user-item bipartite graphs which are mapped to user and item unique identifiers and trained end-to-end in the downstream task).

3.2 The message passing algorithm

Independently on the graph learning strategy adopted (see Section 3.3), the most atomic building block of any graph neural network architecture lies in the *message passing* algorithm. In its most generalized version, such a procedure works by *updating* the latent representation of each node (defined as *ego* node in this context) through the information conveyed by the nodes directly connected to the *ego* one (defined as the *neighbor* nodes). The information from the neighborhood is usually referred to as *messages*, which are *aggregated* and used to update the representation of the *ego* node. Finally, the message passing schema is iteratively applied for L layers, and the various node representations are eventually *combined* to obtain a unique representation for each node. As usually reported in the literature, the message passing algorithm is a generalization of the convolution operation (performed on 2D data such as images) to non-Euclidean data (such as graphs). In the following, a formal definition for each of such steps, namely, *message aggregation*, *node embedding update*, and *layer combination*, is presented and formally described.

3.2.1 Message aggregation

Let $\mathbf{x}_v^{(l)} \in \mathbb{R}^{d_l}$ be the representation of node $v \in \mathcal{V}$ at layer l, with $0 \leq l \leq L$, and d_l the embedding dimension at layer l. We obtain the aggregated messages from the neighbor nodes of v through the following:

$$\mathbf{m}_v^{(l)} = aggregate^{(l)}(\{\mathbf{x}_w^{(l)} \mid w \in \mathcal{N}(v)\}), \tag{3.2}$$

where $\mathbf{m}_v^{(l)} \in \mathbb{R}^{d_l}$ is the aggregated message from the neighbor nodes of v at layer l, $aggregate^{(l)}$ is the aggregation function over the neighbor nodes of v, and $\mathbf{x}_w^{(l)} \in \mathbb{R}^{d_l}$ is the embedding of one of the neighbor nodes of v. For instance, a popular choice for the $aggregate^{(l)}$ function is the element-wise addition:

$$\mathbf{m}_v^{(l)} = \sum_{w \in \mathcal{N}(v)} A_{vw} \mathbf{x}_w^{(l)}, \tag{3.3}$$

where A_{vw} is the entry of the adjacency matrix corresponding to weight of the edge connecting the nodes v and w (if any). The adjacency matrix may be normalized into the symmetric normalized adjacency matrix, especially when it is required to flatten the importance of any neighbor node independently on its degree:

$$\mathbf{m}_v^{(l)} = \sum_{w \in \mathcal{N}(v)} \frac{A_{vw}}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(w)|}} \mathbf{x}_w^{(l)}, \tag{3.4}$$

where in the denominator we are considering the node degree of the ego node and each neighbor node.

3.2.2 Node embedding update

Once the aggregated message from the neighbor nodes has been calculated, the node embedding can be updated to obtain the representation at layer $l+1$:

$$\mathbf{x}_v^{(l+1)} = update^{(l)}(\mathbf{x}_v^{(l)}, \mathbf{m}_v^{(l)}), \tag{3.5}$$

where $\mathbf{x}_v^{(l+1)} \in \mathbb{R}^{d_{l+1}}$ is the updated embedding representation of node v at layer $l+1$, $update^{(l)}$ is the update function for the ego node at layer l, and $\mathbf{x}_v^{(l)}$ is usually included to leverage the current representation of the ego node into the message passing formulation. For instance, a popular choice to design the $update^{(l)}$ function is a neural network where $\mathbf{x}_v^{(l)}$ and $\mathbf{m}_v^{(l)}$ are the two inputs:

$$\mathbf{x}_v^{(l+1)} = \sigma(\mathbf{W}_{ego}^{(l)}\mathbf{x}_v^{(l)} + \mathbf{W}_{neigh}^{(l)}\mathbf{m}_v^{(l)}), \tag{3.6}$$

where $\sigma(\cdot)$ is the activation function of the neural network (e.g., the sigmoid or ReLU), while $\mathbf{W}_{ego}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ and $\mathbf{W}_{neigh}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ are the weights of the neural networks referring to the ego node and the message aggregated through the *aggregate*$^{(l)}$ function, respectively.

3.2.3 Layer combination

The final stage in the message passing algorithm is the combination of all node embedding representations obtained in each layer l. Let $\mathbf{x}_v^{(0)} = \mathbf{x}_v$ be the node representation for the layer 0, which corresponds to the initial embedding representation of the node (e.g., initialized with random values). Then, the layer combination is obtained as:

$$\hat{\mathbf{x}}_v = combine(\{\mathbf{x}_v^{(0)}, \mathbf{x}_v^{(1)}, \dots, \mathbf{x}_v^{(l)}, \dots, \mathbf{x}_v^{(L)}\}), \tag{3.7}$$

where *combine* can be different operations, such as the element-wise addition, the mean, the concatenation, or the last obtained representation is selected:

$$\text{ADD: } \hat{\mathbf{x}}_v = \sum_{0 \le l \le L} \mathbf{x}_v^{(l)}, \tag{3.8}$$

$$\text{MEAN: } \hat{\mathbf{x}}_v = \frac{1}{L} \sum_{0 \le l \le L} \mathbf{x}_v^{(l)}, \tag{3.9}$$

$$\text{CONCAT: } \hat{\mathbf{x}}_v = \{\mathbf{x}_v^{(0)} \mid\mid \mathbf{x}_v^{(1)} \mid\mid \dots \mid\mid \mathbf{x}_v^{(l)} \dots \mid\mid \mathbf{x}_v^{(L)}\}, \tag{3.10}$$

$$\text{LAST: } \hat{\mathbf{x}}_v = \mathbf{x}_v^{(L)}, \tag{3.11}$$

where $\hat{\mathbf{x}}_v$ is the final representation of the node v.

3.2.4 Matrix format and self-loops

The formulations provided above for the message passing algorithm are expressed at node level. However, in most of the cases, the message passing algorithm can be expressed into a more compact matrix format which comprises all graph nodes at once, and the formulation becomes as follows:

$$\mathbf{X}^{(l+1)} = \sigma(\mathbf{X}^{(l)}\mathbf{W}_{ego}^{(l)} + \mathbf{A}\mathbf{X}^{(l)}\mathbf{W}_{neigh}^{(l)}), \tag{3.12}$$

where $\mathbf{X}^{(l)} \in \mathbb{R}^{|\mathcal{V}|\times d_l}$ includes all node embeddings into one matrix. Generally, the *update*$^{(l)}$ function is removed by adding self-loops in the adjacency matrix, namely, the ego node is added to the set of neighbor nodes during the message aggregation operation. The matrix formulation becomes:

$$\mathbf{X}^{(l+1)} = \sigma((\mathbf{A}+\mathbf{I})\mathbf{X}^{(l)}\mathbf{W}^{(l)}), \tag{3.13}$$

where $\mathbf{A}+\mathbf{I}$ is used to add self-loops to the adjacency matrix, and the presence of a single $\mathbf{W}^{(l)}$ matrix indicates that the $\mathbf{W}^{(l)}_{ego}$ and $\mathbf{W}^{(l)}_{neigh}$ are shared into one matrix.

3.3 Popular graph neural network architectures

Depending on the specific strategies and operations adopted for the message passing, the literature recognizes different graph neural networks architectures. In the following, we present and formalize the most popular ones, namely, graph convolutional network (GCN), graph attention network (GAT), and graph isomorphism network (GIN).

3.3.1 Graph convolutional network

The graph convolutional network architecture (GCN) proposed by Kipf et al. [158] is one of the pioneer works in graph neural networks. The layer is defined as:

$$\mathbf{X}^{(l+1)} = \sigma(\tilde{\mathbf{A}}\mathbf{X}^{(l)}\mathbf{W}^{(l)}), \tag{3.14}$$

where $\tilde{\mathbf{A}} = (\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}$ is the symmetric normalized adjacency matrix (with self-loops obtained through the identity matrix $\mathbf{I} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$) and $\sigma(\cdot)$ is the activation function (usually the ReLU).

3.3.2 Graph attention network

The graph attention network architecture (GAT) was first introduced by the work of Velickovic et al. [313], and it is one of the first approaches to leverage attention mechanisms to weight the relative importance of each neighbor node to its corresponding ego node. Specifically, the message passing schema of GAT is defined as:

$$\mathbf{X}^{(l+1)} = \sigma(\Lambda\mathbf{A}\mathbf{X}^{(l)}\mathbf{W}^{(l)}), \tag{3.15}$$

where $\Lambda \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the weight matrix obtained through attention mechanisms. Specifically, by considering the ego node $v \in \mathcal{V}$ along with its neighbor nodes $\forall w \in \mathcal{N}(v)$, the attention mechanism is formulated as:

$$\Lambda_{vw} = \frac{exp(\text{LeakyReLU}(\mathbf{T}[\mathbf{W}\mathbf{x}_v||\mathbf{W}\mathbf{x}_w]))}{\sum_{w\in\mathcal{N}(v)} exp(\text{LeakyReLU}(\mathbf{T}[\mathbf{W}\mathbf{x}_v||\mathbf{W}\mathbf{x}_w]))}, \tag{3.16}$$

where $exp(\cdot)$ is the exponential function, $\mathbf{T} \in \mathbb{R}^{1\times 2d_l}$ is the projection matrix, and $||$ is the concatenation operation.

3.3.3 Graph isomorphism network

By demonstrating that graph neural network architectures such as GCN cannot properly capture the differences between various graph structures, the work by Xu et al. [352] proposes a novel and simple solution, that is as powerful as the Weisfeiler-Lehman graph isomorphism test. For this reason, the architeture is called graph isomorphism network (GIN), and its layer is formulated as:

$$\mathbf{X}^{(l+1)} = mlp((\mathbf{A} + (1+\epsilon)\mathbf{I})\mathbf{X}^{(l)}), \tag{3.17}$$

where $mlp(\cdot)$ is a neural network layer, and ϵ is a constant.

3.4 Tasks in graph representation learning

This last section of the chapter is devoted to presenting three popular tasks which use graph neural networks, namely, (i) node classification, (ii) link prediction, and (iii) graph structure learning. Their formalizations are based upon the *encoder-decoder* framework. In its most generic version, an *encoder* function maps the adjacency matrix and initial node features of a graph to an embedded representation of the nodes:

$$\text{ENCODER: } \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|\times d_0} \rightarrow \mathbb{R}^{|\mathcal{V}|\times d_L}. \tag{3.18}$$

Given the way it is formulated, the *encoder* function is nothing but any message passing procedure iterated over L propagation layers, namely, any presented graph neural network architecture from above.

As for the *decoder* function, it takes the encoded representation of the nodes as input, and produces some outputs which depend on the specific task we are considering. Such an output may be a probability vector over the set of possible classes $\mathcal{C}$ (i.e., node classification), a similarity score between two node embeddings (i.e., link prediction), or a learned version of the adjacency matrix (i.e., graph structure learning):

$$\text{DECODER: } \mathbb{R}^{|\mathcal{V}|\times d_L} \underbrace{\left[\times\mathbb{R}^{|\mathcal{V}|\times d_L}\right]}_{\text{LP}} \to \underbrace{\mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}}_{\text{GSL}} \times \underbrace{\mathbb{R}^{|\mathcal{V}|\times|\mathcal{C}|}}_{\text{NC}} \times \underbrace{\mathbb{R}^{+}}_{\text{LP}}, \tag{3.19}$$

where we abbreviate link prediction, graph structure learning, and node classification through LP, GSL, and NC, respectively. In the following, we describe the three selected tasks regarding graph representation learning, by re-casting each of them into the encoder-decoder framework we just presented.

3.4.1 Node classification

Node classification is probably one of the primary and most popular task to benchmark the performance of graph neural networks. As any classification task in machine learning, node classification involves the training of a graph neural network model to predict the correct class each node in the graph belongs to. Let $\mathbf{y}_v \in \mathbb{R}^{|\mathcal{C}|}$ be the one-hot-encoding vector indicating to which class node $v \in \mathcal{V}$ belongs to, where $y_v[c]$ is 1 if v belongs to class $c \in \mathcal{C}$, 0 otherwise.

In terms of *encoder* function, it maps the adjacency matrix and the initial node features to embedded node representations:

$$\text{ENCODER: } \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|\times d_0} \to \mathbb{R}^{|\mathcal{V}|\times d_L}, \tag{3.20}$$

whose formulation is a graph neural network:

$$\text{ENCODER}(\mathbf{A},\mathbf{X}) = \hat{\mathbf{X}}. \tag{3.21}$$

As for the *decoder* function, it maps the embedded node representations to probability vectors over the set of possible classes $\mathcal{C}$:

$$\text{DECODER: } \mathbb{R}^{|\mathcal{V}|\times d_L} \to \mathbb{R}^{|\mathcal{V}|\times|\mathcal{C}|}, \tag{3.22}$$

whose formulation is:

$$\text{DECODER}(\text{ENCODER}(\mathbf{A},\mathbf{X})) = \text{DECODER}(\hat{\mathbf{X}}) = \hat{\mathbf{Y}}, \tag{3.23}$$

where $\hat{\mathbf{Y}}$ is the predicted vector probability over the set of classes $\mathcal{C}$ for all nodes.

The loss function for node classification is the negative log-likelihood loss function:

$$\mathcal{L} = -log(softmax(\hat{\mathbf{Y}}, \mathbf{Y})), \tag{3.24}$$

where $log(\cdot)$ and $softmax(\cdot)$ are the logarithm and softmax functions, respectively, while $\mathbf{Y}^{|\mathcal{V}|\times|\mathcal{C}|}$ is the one-hot-encoding vector indicating to which class each node belongs to, respectively. Differently from other classification tasks in machine learning, node classification is often referred to as a *semi-supervised* task because nodes in the test set (whose label/class is not known during the training) are still exploited in the message passing procedure. However, their are not used in the calculation and minimization of the loss function. In such cases, nodes in the test set are defined as *transductive.* On the contrary, when test nodes are not used during the training phase at all, such nodes are defined as *inductive.*

3.4.2 Link prediction

Link prediction is another popular task where graph neural networks are widely exploited. The task involves the prediction, in a graph structure, of the existence of an edge between pairs of nodes.

In the case of link prediction, the *encoder* function maps the adjacency matrix and the initial node representations to the nodes embedded representations in the latent space:

$$\text{ENCODER: } \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|\times d_0} \to \mathbb{R}^{|\mathcal{V}|\times d_L}, \tag{3.25}$$

As already seen for node classification, the encoder function is formulated as:

$$\text{ENCODER}(\mathbf{A}, \mathbf{X}) = \hat{\mathbf{X}}. \tag{3.26}$$

A *decoder* function, paired with the *encoder* one, is aimed at reconstructing some graph structure properties, such as which are the neighbor nodes of a given ego node:

$$\text{DECODER: } \mathbb{R}^{|\mathcal{V}|\times d_L} \times \mathbb{R}^{|\mathcal{V}|\times d_L} \to \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|} \times \mathbb{R}^{+}. \tag{3.27}$$

For instance, we apply the *decoder* function to predict whether there exists an edge between each pair of nodes in the graph (i.e., we are approximating the adjacency matrix):

$$\text{DECODER}(\text{ENCODER}(\mathbf{X}), \text{ENCODER}(\mathbf{X})) = \text{DECODER}(\hat{\mathbf{X}}, \hat{\mathbf{X}}) = \hat{\mathbf{A}}, \tag{3.28}$$

where $\hat{\mathbf{A}}$ is the approximated version of the adjacency matrix. A possible way of modeling the decoder function is through the matrix factorization paradigm, which calculates the similarity score of two entities (nodes) through the inner product of their embedded representations. As the message passing allows to incorporate the information conveyed by the neighbor nodes into the ego node at multiple layers, the inner product between two ego nodes provides also an indication of how the neighbor nodes of the two ego nodes are similar (i.e., we are reconstructing the graph structural properties). Thus, in the case of inner-product, we have:

$$\text{DECODER}(\hat{\mathbf{X}}, \hat{\mathbf{X}}) = \hat{\mathbf{X}}^{\top}\hat{\mathbf{X}}. \tag{3.29}$$

In such a scenario, the loss is a *pairwise* one, and may be the mean squared error:

$$\mathcal{L} = ||\hat{\mathbf{X}}^{\top}\hat{\mathbf{X}} - \mathbf{A}||_2^2. \tag{3.30}$$

Note that we are, as a common practice, the adjancecy matrix is masked during the training phase, so that only some edges are known, while the remaining ones are used to test the performance of the trained model. Moreover, the training set may contain both positive (i.e., existing) and negative (i.e., non-existing) edges. The negative edges are obtained through *negative sampling* strategies, such as the one adopted in Bayesian personalized ranking (BPR) in recommendation [258].

3.4.3 Graph structure learning

Finally, we take into account the task of learning and/or refining the structure of a graph, namely, graph structure learning.

As observed for the other tasks in graph representation learning, graph structure learning involves an *encoder* function defined as follows:

$$\text{ENCODER: } \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|\times d_0} \to \mathbb{R}^{|\mathcal{V}|\times d_L}, \tag{3.31}$$

which is a graph neural network:

$$\text{ENCODER}(\mathbf{A}, \mathbf{X}) = \hat{\mathbf{X}}. \tag{3.32}$$

The *decoder* function works by generating an updated/refined version of the adjacency matrix, starting from the embedded node representations obtained from the *encoder*.

Its signature is:

$$\text{DECODER: } \mathbb{R}^{|\mathcal{V}| \times d_L} \rightarrow \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}. \tag{3.33}$$

In this respect, the literature mainly recognizes three families of strategies to model the *decoder* function. The first involves metric-wise approaches, which leverage similarity metrics to estimate the edge weight between pairs of nodes, such as the inner-product or the cosine similarity of node embeddings:

$$\underbrace{\text{DECODER}}_{\text{inner-product}}\text{: } \sigma(\hat{\mathbf{X}}^\top \hat{\mathbf{X}}), \tag{3.34}$$

$$\underbrace{\text{DECODER}}_{\text{cosine}}\text{: } cos(\hat{\mathbf{X}} \odot \mathbf{W}, \mathbf{X} \odot \mathbf{W}), \tag{3.35}$$

where $\sigma(\cdot)$ is any activation function (e.g., the sigmoid), while $cos(\cdot)$ is the cosine similarity function, and $\mathbf{W}$ are trainable weights (e.g., obtained from the edge features in the graph). The second type of approaches is defined as neural approaches, and the graph attention network layer belongs to this family (refer again to Section 3.3.2). Finally, the third group of approaches works by learning the entries of the adjacency matrix as free parameters.

Generally, graph structure learning is performed as a side task along with a main task (such as node classification). For this reason, the loss function involves two components, namely, a main task loss, and a regularization loss:

$$\mathcal{L} = \mathcal{L}_{task} + \lambda \mathcal{L}_{reg}(\hat{\mathbf{A}}, \hat{\mathbf{X}}). \tag{3.36}$$

The regularization term may assume different formulations depending on the graph properties one wants to optimize, such as the sparsity of the adjacency matrix, or the smoothness.

Chapter 4

Formalizing multimedia recommendation

In this chapter, we first provide a comprehensive literature review of multimodal approaches for multimedia recommendation from the last eight years. Second, we outline the theoretical foundations of a multimodal pipeline for multimedia recommendation by identifying and formally organizing recurring solutions/patterns. Third, we demonstrate its rationale by conceptually applying it to selected state-of-the-art approaches in multimedia recommendation. Finally, we highlight the significant unresolved challenges in multimodal deep learning for multimedia recommendation. Our primary aim is to provide guidelines for designing and implementing the next generation of multimodal approaches in multimedia recommendation.

4.1 Motivations

Our experience of daily life is intrinsically *multimodal*. We interact with objects surrounding us through our five senses. For instance, watching a movie can involve three senses (i.e., modalities): we watch it (*visual* modality) while listening to the dialogues (*audio* modality) and possibly reading its subtitles (*textual* modality). Multimodal learning has been one of the hot topics in deep learning for some years now, addressing applicative domains such as medical imaging [34, 109, 129, 303], autonomous driving [44, 152, 350, 400], speech/emotion recognition [173, 198, 241, 244], multimedia retrieval [53, 135, 136, 174], and, only recently, multimodal large language modelling [368]. Given the success and popularity it has encountered, some works have tried to outline, categorize, and formalize the core concepts behind multimodality in deep learning [28, 29, 227]. Remarkably, the literature recognizes five steps and challenges when designing a multimodal deep learning pipeline [29]: *representation*, *translation*, *alignment*, *fusion*, and *co-learning*.

Similarly to the cited domains and applications, approaches in multimedia recommendation have been shown to effectively apply multimodal deep learning techniques to the recommendation task. The idea is to model users' and items' profiles through the different modalities and suitably capture the multi-faceted nature of their interconnections. Recent works in the literature have brought multimodality to multimedia recommendation [189, 269, 309, 404] tackling (just to mention a few) micro-video recommendation [45, 67, 339], food recommendation [171, 222, 321], outfit fashion compatibility [65, 361, 380], and artist/song recommendation [70, 236, 312]. However, and differently from the other outlined domains and scenarios, recommendation lacks a *shared* theoretical formalization to align the multimedia recommendation problem with the same formal pipeline proposed in multimodal deep learning [28, 29, 227].

For these reasons, in this chapter, we first review the most popular and recent state-of-the-art approaches in multimedia recommendation. Indeed, it emerges that three main design choices are involved when proposing novel multimedia recommender systems leveraging multimodality: (i) ***Which*** modalities to suitably describe the user/item input data; (ii) ***How*** to extract and process meaningful multimodal representations; (iii) ***When*** to integrate and inject multimodal data into the training/inference steps. While observing that many multimedia recommendation approaches are rarely aligned on the techniques to adopt for (i), (ii), and (iii), we maintain this could limit the future development of novel solutions in the field. This is true since each work claims to advance with respect to the state-of-the-art but it becomes cumbersome to distinguish which conceptual *strategies* are contributing the most [206]. Thus, inspired by the multimodal pipeline formalized in multimodal deep learning [28, 29, 227], we try to align the same schema with the three design choices recognized above. Our objective is to define a conceptual and theoretical schema that uses multimodality to encompass and summarize the most diffused solutions/patterns in the multimedia recommendation literature. To the best of our knowledge, this represents the first attempt that, differently from similar works in the literature [189, 404], ***formalizes*** multimedia recommendation through the core concepts theorized in multimodal deep learning [28, 29, 227].

To sum up, our contributions are:

1. We review existing works in multimedia recommendation adopting multimodal learning techniques, highlighting common and different architectural choices; in this respect, we categorize the reviewed papers according to the type of

multimodal input (i.e., ***What***), the technique for features processing (i.e., ***How***), and the moment to integrate modalities (i.e., ***When***).

2. On such basis, and following the related literature on multimodal deep learning, we revisit the multimedia recommendation task under the lens of multimodal deep learning; by mapping the multimodal pipeline outlined in [28, 29, 227] to the threefold categorization from above, we provide the general formulations for a formal schema involving three steps: multimodal input data, multimodal feature processing, and multimodal feature fusion.
3. First, we select four multimodal approaches from the recent literature spanning various domains and scenarios in multimedia recommendation; then, we show how the proposed multimodal schema conceptually applies to the selected models in all the steps of the pipeline, thus proving its effectiveness.
4. Driven by the previous findings, we outline technical and conceptual challenges.

We release a GitHub repository with all the reviewed papers: https://github.com/sisinflab/Formal-MultiMod-Rec.

4.2 Literature review

In this section, we present a literature review on recent multimodal applications for the task of multimedia recommendation. Table 4.1 reports 43 papers collected from the proceedings of top-tier conferences and journals over the last eight years. A careful review and analysis aimed at outlining recurrent schematic and observed patterns suggests categorizing the retrieved papers according to three key questions:

- ***Which*** modalities to choose for the input data?
- ***How*** to process multimodal features in terms of feature extraction and representation?
- ***When*** to fuse the different modalities to integrate them into the final recommendation framework?

To collect all reviewed papers, we also include a public GitHub repository[1] to access their direct DOIs. We intend to update this repository with the most recent works leveraging multimodality for multimedia recommendation.

[1]https://github.com/sisinflab/Formal-MultiMod-Rec.

Table 4.1 Overview of the core questions which arise when modelling a multimedia recommender system based upon multimodality, as observed in the most updated literature. HFE: Handcrafted Feature Extraction, TFE: Trainable Feature Extraction, MMR: Multimodal Representation.

Papers	**Year**	**Modalities (*Which?*)**			**Feature Processing (*How?*)**					**Fusion (*When?*)**	
		Visual	*Textual*	*Audio*	HFE	TFE		MMR		*Early*	*Late*
						Pretrained	*End-to-End*	*Joint*	*Coordinate*		
Ferracani et al. [98]	2015	✓	✓			✓		✓			
Jia et al. [144]	2015	✓	✓		✓			✓			
Li et al. [175]	2015	✓		✓	✓			✓			
Nie et al. [232]	2016	✓	✓		✓				✓	✓	
Chen et al. [63]	2016	✓	✓		✓			✓			
Han et al. [120]	2017	✓	✓				✓		✓	✓	
Oramas et al. [236]	2017		✓	✓			✓		✓	✓	
Zhang et al. [385]	2017	✓	✓				✓		✓	✓	
Ying et al. [369]	2018	✓	✓			✓		✓			
Wang et al. [317]	2018	✓	✓			✓		✓			
Liu et al. [181]	2019	✓	✓			✓		✓			
Chen et al. [66]	2019	✓	✓			✓					
Wei et al. [339]	2019	✓	✓	✓		✓			✓	✓	
Cheng et al. [69]	2019	✓	✓			✓					
Dong et al. [91]	2019	✓	✓			✓			✓	✓	
Chen et al. [65]	2019	✓	✓			✓		✓			
Yu et al. [373]	2019	✓	✓			✓	✓	✓	✓	✓	
Cui et al. [78]	2020	✓	✓			✓			✓		✓
Wei et al. [338]	2020	✓	✓	✓		✓			✓	✓	
Sun et al. [293]	2020	✓	✓			✓		✓			
Chen et al. [57]	2020	✓	✓			✓		✓			
Min et al. [222]	2020	✓	✓	✓		✓		✓			
Shen et al. [277]	2020	✓	✓		✓	✓			✓	✓	
Yang et al. [361]	2020	✓	✓			✓			✓		✓
Tao et al. [305]	2020	✓	✓	✓		✓			✓	✓	
Yang et al. [359]	2020	✓	✓			✓			✓	✓	
Sang et al. [270]	2021	✓	✓			✓			✓	✓	
Liu et al. [193]	2021	✓	✓	✓		✓			✓	✓	
Zhang et al. [382]	2021	✓	✓			✓			✓	✓	
Vaswani et al. [312]	2021		✓	✓		✓			✓	✓	
Lei et al. [171]	2021	✓	✓	✓		✓	✓				
Wang et al. [321]	2021	✓	✓			✓		✓			
Zhan et al. [380]	2022	✓	✓			✓		✓			
Wu et al. [341]	2022	✓	✓			✓			✓	✓	
Yi et al. [365]	2022	✓	✓			✓			✓		✓
Yi et al. [366]	2022	✓	✓	✓		✓			✓		✓
Liu et al. [194]	2022	✓	✓			✓			✓	✓	
Mu et al. [223]	2022	✓	✓			✓			✓	✓	
Chen et al. [55]	2022	✓	✓	✓		✓			✓	✓	
Zhou et al. [406]	2022	✓	✓			✓			✓		✓
Wang et al. [320]	2023	✓	✓	✓		✓			✓	✓	
Wei et al. [336]	2023	✓	✓	✓		✓			✓		✓
Zhou et al. [407]	2023	✓	✓			✓			✓		✓

4.2.1 *Which* modalities?

In multimedia recommendation scenarios, input data generally comes in at least two of the three most common modalities in literature, namely *Visual*, *Textual*, and *Audio* modalities. As evident from the collected papers, the vast majority of works consider the visual and textual modalities, which mainly refer to product images and descriptions (e.g., [78, 91, 120, 181, 380, 382]), respectively, while fewer examples leverage such modalities to describe video frames and captions (e.g., [144, 270, 321]) or users' social media interactions through uploaded photographs together with texts (e.g., [63, 359, 385]). Another emerging trend from the literature is that audio is by far the most underrepresented modality, and it is usually coupled with the textual one to describe music in the form of audio signals and songs' descriptions (e.g., [236, 312]). Conversely, the related literature shows that the audio modality is frequently exploited for video input data (e.g., [305, 320, 338, 339, 366]) which is also the unique scenario involving all modalities.

The observed disparity in data modalities is not only linked to the specific task the various approaches address (e.g., product, song, or micro-video recommendation) but it is also found in each modality's different availability. In this respect, for example, datasets collecting user-item interactions on e-commerce platforms (e.g., the Amazon reviews dataset or IQON300) are more easily accessible than the ones involving social media videos. For instance, one may consider that a version of the TikTok dataset (introduced in [339]) has been made available with pre-trained multimodal features involving visual, audio, and textual modalities only recently [336]. This modality *misalignment* is among the most discussed challenges in the community, so we decide to dedicate a section to it later (refer to Section 4.5.1).

4.2.2 *How* to process modalities?

Once modalities have been selected for data inputs, two primary operations usually get involved in processing the multimodal data to be fed into the recommender system. First, high-level features are extracted from each of the available modalities. Interestingly, early approaches adopt handcrafted feature extraction (HFE) strategies (e.g., color histograms) as described in [63, 144, 175, 232]. However, with the outbreak and the increasing popularity encountered by deep learning and deep neural models for image and text classification, object detection, and speech recognition, trainable feature extractors (TFE) soon became the de facto standard in the learning of latent features from the input data. In this respect, the literature [82] indicates that the common approach is to use the activation of one of the final hidden layers of deep neural networks. For instance, the authors of [193, 194, 222, 293, 317, 336, 373] exploit features extracted from deep networks. Furthermore, we categorize TFE strategies based on the use of *Pretrained* deep networks and *End-to-End* learned models. The former refer to the possibility of transferring the learned knowledge of already-trained deep networks to different domains, tasks, and datasets (e.g., see [300]), whereas the latter usually leverage custom deep neural networks trained in the downstream recommendation task. As evident from the collected papers, the pre-trained solution (e.g., [57, 66, 91, 98, 270, 277, 338, 369]) widely surpasses the end-to-end one (e.g., [120, 236, 385]) in terms of popularity, as the adoption of ready-to-use embedded features obtained from state-of-the-art deep learning models represents a more efficient and convenient approach than performing computationally-expensive and data-eager trained feature extractions. Nevertheless, an argument might be made that using features extracted through models already trained on different datasets and tasks could limit their expressiveness regarding the actual multimedia recommendation task. For this reason, we deepen into the issue in Section 4.5.2.

The second operation involved in the feature processing phase regards the implementation of a multimodal representation (MMR) solution to establish relations among the extracted modalities. We recognize two main approaches, namely, either combining all modalities so that they belong to a unique representation (*Joint*) or keeping them separated to leverage the different influence they may have on recommendation (*Coordinate*). From the collected papers, it follows that both the former (e.g., [55, 65, 98, 223, 293, 369, 380]), and the latter (e.g., [232, 339, 361, 365, 382]) are almost equally preferred; however, the coordinate multimodal representation is slightly more popular as learning different representations for each involved modality may help unveil the specific contribution it brings to the final personalized recommendation. Indeed, this could support *explainability*, which is among the hottest topics in the community [391], and especially in multimedia recommendation scenarios, where user-item interactions may, by nature, be driven by non-evident and sometimes contrasting users' preferences and tastes [60, 66, 338]. Finally, the authors from [66, 69] do not integrate any multimodal representation approach since they exploit multimodality only for the optimization of the loss but not to predict user-item preferences.

4.2.3 *When* to fuse modalities?

The last stage in the multimodal pipeline deals with the fusion of the different processed modalities so that they can be eventually integrated into the recommendation outcome as a single representation of multiple *coordinated* modalities. This process may take place *before* or *after* the prediction of the user-item preference score. On this basis, the former and the latter approaches are usually known as *Early* (e.g., [312, 339]) and *Late* (e.g., [78, 365]) fusion, respectively. It is worth pointing out that some solutions recognize a third strategy (i.e., *Hybrid* fusion) that combines the two versions mentioned above, but for the sake of simplicity, we decide to categorize the works performing this kind of multimodal fusion as a particular case of late fusion. Additionally, we recognize that several approaches from the literature do not provide a precise differentiation between joint multimodal representation and early fusion. To better clarify this technical aspect, we propose to consider *fusion* as an optional operation that takes place after the feature processing phase only in the case of *coordinate* multimodal representation (you may refer to Section 4.5.3). Indeed, as evident from the table, *Joint* multimodal representation and *Early/Late* fusion never occur in the same approach. What is more, we observe that early fusion, employed, for instance, in [55, 91, 194, 223, 232, 270, 277, 341, 382, 385], is more popular than late fusion, used in [78, 336, 361, 365, 366, 406, 407]. Motivating this tendency is an unanswered research question that we leave as a possible open issue to impact the design of recommender systems leveraging multimodality. In this respect, you may refer to Section 4.5.4 for our discussion on the current challenges about modality fusion.

4.2.4 Similar works

For the sake of completeness, we review the current literature works that provide similar contributions to ours to outline the main differences. As already mentioned, pioneer works such as [28, 29, 227] introduce and formalize (for the first time) the core concepts and ideas behind the field of multimodal deep learning. After that, the recent years have seen a growing interest in systematically reviewing and schematizing techniques for multimodal fusion [100], spanning different application domains such as medicine [306], conversational artificial intelligence [297], and visual content syntesis [393], up to addressing complex and novel machine learning strategies including meta-learning [203]. Although the cited works share similar rationales to ours, their focus is more general (e.g., deep learning) or heterogeneous (e.g., medicine) with respect to the multimedia recommendation task.

In the recommendation domain, the study presented in [222] is among the closest and most influential works to our proposal in the intention of introducing a unified framework for food recommendation which leverages the concept of multimodality; however, the work is different from ours in that: (i) it only addresses the task of food recommendation, and (ii) it does not provide either mathematical formalizations analysis of the proposed multimodal pipeline. Furthermore, it is worth recalling two surveys regarding the topic of multimodal recommender systems [189, 404] on arXiv at the moment this thesis is written. Among the two, the work presented in [404] shows the major similarities to our proposal, especially when recognizing a multimodal pipeline for multimedia recommendation. Nevertheless, what we propose in this chapter stands out for the following novel contributions: (i) we systematically follow the multimodal pipeline outlined in [28, 29, 227] in the attempt to adapt it to the three main questions arising in the multimedia recommendation literature, namely, ***Which?***, ***How?***, and ***When?***; (ii) we provide mathematical formalizations for each step of the proposed multimodal pipeline to sketch a formal schema for the next generation of multimodal approaches addressing multimedia recommendation; (iii) we identify a wider set of challenges regarding each step in the multimodal pipeline.

4.3 A formal multimodal schema for multimedia recommendation

As previously outlined, the literature shows recurrent schematic patterns in adopting multimodal techniques for the task of multimedia recommendation. However, when considering the latest solutions in the field (Section 4.2) it appears evident that, differently from what happens for other applicative domains in machine learning, such approaches do not seem to follow any shared and officially recognized formal schema aligned with the principles of multimodal deep learning [28, 29, 227]. To sort things out, in this section, we propose to formally revisit multimedia recommendation under the lens of multimodal deep learning (Figure 5.2). First, we formalize the standard recommendation task. Then, we theoretically give answers to the core questions previously outlined, namely: ***Which*** multimodal input data to adopt, ***How*** to extract multimodal features and set relationships among them, and ***When*** to fuse modalities. Finally, we specify the multimedia recommendation task through multimodality. In the following, we use the **bold** notation only when we explicitly define a vector for which we indicate its elements (i.e., scalars or other vectors).

4.3.1 Classical recommendation task

Despite we already introduced and described the recommendation task in Section 2.1.2, we recall and expand the main notions in the following. We consider users, items, and user-item interactions as the inputs to the recommender system. We denote with $u \in \mathcal{U}$, $i \in \mathcal{I}$, and $r \in \mathcal{R}$ a user, an item, and a user-item interaction, respectively. To ease the notation, we say $x \in \mathcal{X}$ is a general input to the system, with $\mathcal{X} = \mathcal{U} \cup \mathcal{I} \cup \mathcal{R}$. Given a set of input data $\mathcal{X}$, and defined $\rho(\cdot)$ as the preference score prediction function, a recommender system aims to build a top@k list of items maximizing the following

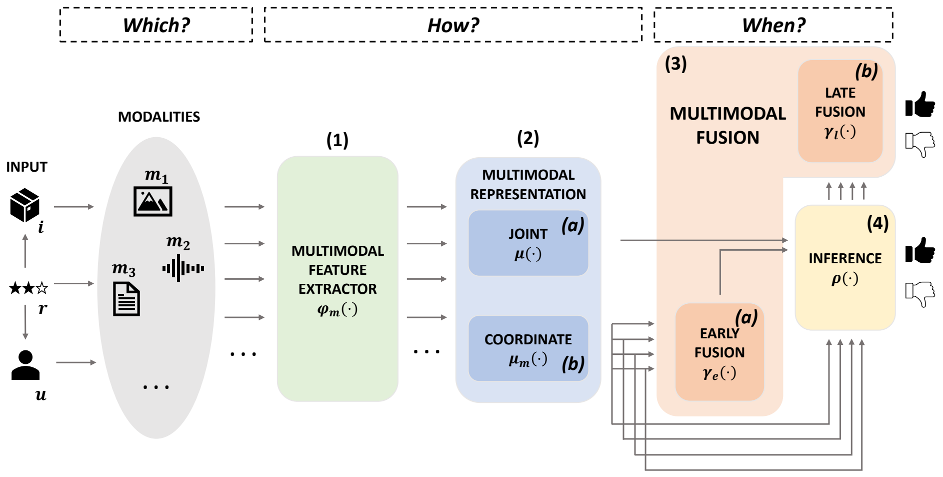

Which?
How?
When?
INPUT
i
r
u
MODALITIES
m_1
m_2
m_3
(1)
MULTIMODAL FEATURE EXTRACTOR $\varphi_m(\cdot)$
(2)
MULTIMODAL REPRESENTATION
JOINT $\mu(\cdot)$ (a)
COORDINATE $\mu_m(\cdot)$ (b)
(3)
MULTIMODAL FUSION
(b) LATE FUSION $\gamma_l(\cdot)$
(a) EARLY FUSION $\gamma_e(\cdot)$
(4)
INFERENCE $\rho(\cdot)$

Fig. 4.1 Our multimodal schema for multimedia recommendation. After (1) a modality-aware feature extraction, the extracted features may be either directly represented into a unique latent space ($2a$) or projected into a different latent space for each modality ($2b$). While in the former case, the multimodal representation is used to produce a prediction (4), in the latter case, all modalities must undergo a fusion phase (3). In the early fusion ($3a$), we produce a final representation that is used for prediction (4). Otherwise, we first produce a different prediction for each modality (4), and then we fuse them (late fusion) into a single predicted value ($3b$).

posterior probability ($prob$):

$$\hat{\Theta}_\rho = \underset{\mathbf{\Theta}_\rho}{\operatorname{argmax}}\, prob(\mathbf{\Theta}_\rho \mid \mathcal{X}), \tag{4.1}$$

where $\mathbf{\Theta}_\rho = [\theta_\rho^{(0)}, \theta_\rho^{(1)}, \ldots, \theta_\rho^{(|\mathcal{W}_\rho|-1)}]$ is the vector collecting all weights for the inference function $\rho(\cdot)$, $\mathcal{W}_\rho = \{\theta_\rho^{(0)}, \theta_\rho^{(1)}, \ldots, \theta_\rho^{(|\mathcal{W}_\rho|-1)}\}$ is the set of such weights, and $|\mathcal{W}_\rho|$ its cardinality. For instance, in the case of latent factor models (e.g., matrix factorization [160]), the set of trainable weights $\mathcal{W}_\rho$ involves the user and item embeddings.

4.3.2 Multimodal input data

As shown in Figure 5.2, the first step of our multimodal schema is to identify input modalities. A common list of modalities for each input data (i.e., user, item, user-item interaction) in multimedia scenarios may be defined as follows:

- visual (**v**), e.g., images, video frames;
- textual (**t**), e.g., image captions, video subtitles, song lyrics, reviews;
- audio (**a**), e.g., songs, podcasts, movie soundtracks.

Formally, we define $m \in \mathcal{M}$ as an admissible modality for the system (i.e., $\mathcal{M} = \{\texttt{v}, \texttt{t}, \texttt{a}\}$). We should mention that data may come with all such modalities or just a subset. For instance, videos from video streaming platforms (such as Netflix or Amazon Prime Video) have frames ($\texttt{v}$), subtitles and/or descriptions ($\texttt{t}$), and an audio track and/or soundtrack ($\texttt{a}$). Similarly, e-commerce platforms (such as Amazon or Zalando) sell products that may come with photographs ($\texttt{v}$) and reviews which stand for the textual feedback users express towards those products ($\texttt{t}$).

Let $x \in \mathcal{X}$ be an input to the recommender system, whose set of available modalities is indicated as $\mathcal{M}_x \subseteq \mathcal{M}$. We represent the *content* data of input x in modality m as $c_x^{(m)}$, with $m \in \mathcal{M}_x$, and the vector of content data for input x in all modalities as $\mathbf{c}_x$. Concerning the examples from above, a video item x may be described through three modalities (i.e., $\mathcal{M}_x = \{\texttt{v}, \texttt{a}, \texttt{t}\}$) and, for example, its visual content data (a frame) is an RGB image indicated as $c_x^{(\texttt{v})}$. Similarly, a fashion item x may be described through two modalities (i.e., $\mathcal{M}_x = \{\texttt{v}, \texttt{t}\}$) and, for example, its textual content data (the description) is a set of words indicated as $c_x^{(\texttt{t})}$.

4.3.3 Multimodal feature processing

As in Figure 5.2, multimodal inputs are processed to be transferred into a low-dimensional representation. This step runs through a multimodal feature extractor and a component that constructs a multimodal feature representation.

Feature extraction

Content input data is generally not exploitable as it is in a recommender model (e.g., the matrix of pixels from an image is not meant to be directly integrated into a recommender). Hence, our schema introduces a *Feature Extractor* (FE) component to extract features, which should follow two principles, being (i) *high-level* (i.e., meaningful for the recommender system) and (ii) *functional* to the final recommendation task. Indeed, choosing the most suitable feature extractor for each modality may affect the performance.

Let $c_x^{(m)}$ be the content data for input x in modality $m \in \mathcal{M}_x$. Then, let $\varphi_m(\cdot)$ be the feature extractor function for the modality m. We define the feature extraction process in the modality m as:

$$\overline{c}_x^{(m)} = \varphi_m(c_x^{(m)}) \quad \forall m \in \mathcal{M}_x, \tag{4.2}$$

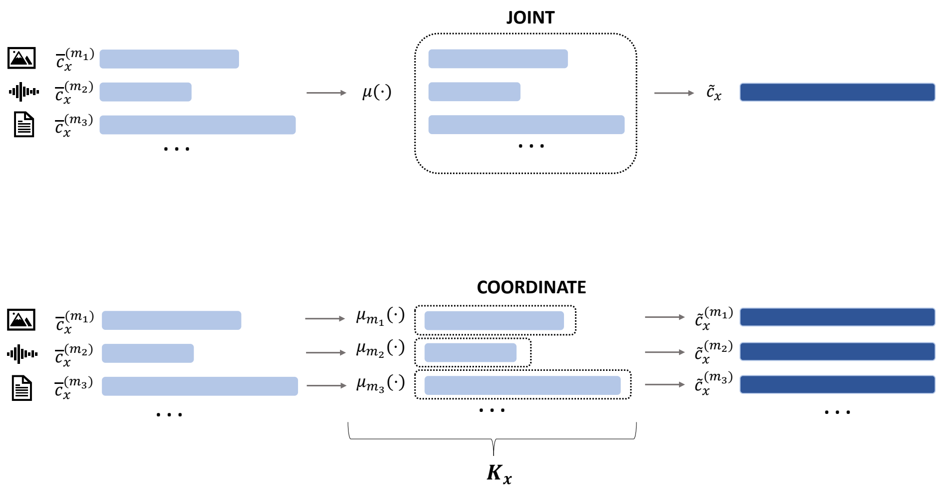


Fig. 4.2 A visual representation of *Joint* and *Coordinate* multimodal representation (above and below, respectively).

where $\bar{c}_x^{(m)}$ is the extracted feature for input x in modality m. We use the notation $\bar{\mathbf{c}}_x = [\bar{c}_x^{(0)}, \bar{c}_x^{(1)}, \dots, \bar{c}_x^{(|\mathcal{M}_x|-1)}]$ to refer to the vector of extracted features for input x in all modalities. Generally speaking, $\varphi_m(\cdot)$ may refer either to a handcrafted extractor, HFE (e.g., SIFT and color histogram for visual, and MFCCs for audio), or to a trainable extractor, TFE (e.g., deep learning-based models such as CNNs for visual, audio, and textual). In the latter case, $\varphi_m(\cdot)$ can be either pre-trained or trained end-to-end along with the recommender system.

Multimodal representation

Once high-level features have been extracted from each modality of the input data, the next step is to design a *Representation* strategy to handle the relationships among modalities and eventually inject such data into the recommender system. As shown in Section 4.2, the literature follows two main approaches: *Joint* and *Coordinate* (Figure 4.2). The former relies on projecting multimodal features into a shared latent space to produce a unique final representation (e.g., concatenation is usually the simplest approach). Conversely, the latter involves adopting a different latent space for each modality, with the possibility of setting specific constraints among modalities that are expressed, for instance, through similarity metrics. In the following, we mathematically formalize the two strategies.

Joint representation. Let $\bar{\mathbf{c}}_x$ be the vector of extracted features for input x in all modalities. In the case of *Joint* representation, we assume $\mu(\cdot)$ is the function to produce the multimodal representation of the extracted features. Thus:

$$\tilde{c}_x = \mu(\overline{\mathbf{c}}_x), \tag{4.3}$$

where $\tilde{c}_x$ is the multimodal representation for input x.

Coordinate representation. Let $\overline{c}_x^{(m)}$ be the extracted feature for input x in modality $m \in \mathcal{M}_x$. In the case of *Coordinate* representation, we assume $\mu_m(\cdot)$ is the multimodal representation function for modality m, and let $\mathcal{K}_x$ be a set of constraints on multimodal representations of input x. Thus, we say:

$$\tilde{c}_x^{(m)} = \mu_m(\overline{c}_x^{(m)}) \textbf{ subject to } \mathcal{K}_x, \textbf{ with } |\mathcal{K}_x| \geq 0, \tag{4.4}$$

where $\tilde{c}_x^{(m)}$ is the coordinate multimodal representation for input x in modality m. Note that in Equation (4.4) we denote with $\tilde{\mathbf{c}}_x = [\tilde{c}_x^{(0)}, \tilde{c}_x^{(1)}, \dots, \tilde{c}_x^{(|\mathcal{M}_x|-1)}]$ the vector of coordinate multimodal representations for input x in all modalities.

4.3.4 Multimodal feature fusion

As an optional third step, when *Coordinate* representation is used, our multimodal schema allows an additional *Fusion* step to combine all produced multimodal representations. In the following, we describe the inference step in the two cases of *Early* and *Late* fusion.

Early fusion. Let $\tilde{\mathbf{c}}_x$ be the vector of coordinate multimodal representations for input x in all modalities. Then, let $\gamma_e(\cdot)$ be the function for *Early* fusion. We generate the multimodal representation for input x as:

$$\tilde{c}_x = \gamma_e(\tilde{\mathbf{c}}_x). \tag{4.5}$$

Note that after applying Equation (4.5), everything we describe in the following also applies to *Joint* representation. We obtain the predicted output $\hat{y}$ for input x as:

$$\hat{R} = \rho(\tilde{c}_x) \tag{4.6}$$

Late fusion. Let $\tilde{c}_x^{(m)}$ be the coordinate multimodal representation for input x in modality $m \in \mathcal{M}_x$. We first predict the different output values for each modality as:

$$\hat{R}^{(m)} = \rho(\tilde{c}_x^{(m)}) \quad \forall m \in \mathcal{M}_x. \tag{4.7}$$

Let $\hat{\mathbf{y}}$ be the vector of multimodal predicted outputs in all modalities. If we denote $\gamma_l(\cdot)$ as the function for *Late* fusion, we finally aggregate (fuse) all modality-aware predictions:

$$\hat{R} = \gamma_l(\hat{\mathbf{R}}). \tag{4.8}$$

Whatever the type of *Fusion*, the literature shows that various works perform this operation differently, from more straightforward solutions such as concatenation and element-wise addition, multiplication, or average to more refined techniques (i.e., neural-based ones, like attention mechanisms). Note that we consider *Late* fusion also when multimodal representations are exploited for some specific components of the loss function; indeed, in such settings, multimodal fusion does not occur even until the very last stage of the recommendation pipeline (i.e., the calculation and optimization of the loss function).

4.3.5 Multimodal recommendation task

Let $\mathcal{W}_\varphi$, $\mathcal{W}_\mu$, and $\mathcal{W}_\gamma$ be the sets of the additional model trainable weights from (i) feature extraction, (ii) multimodal representation, and (iii) multimodal fusion, respectively. Note that they could be empty, as the correspondent functions may be non-trainable. Then, given the set of multimodal input data $\mathcal{X}$, we extend Equation (4.1):

$$\hat{\Theta} = \underset{\mathbf{\Theta}}{\operatorname{argmax}}\, prob(\mathbf{\Theta}|\mathcal{X}), \tag{4.9}$$

where $\mathbf{\Theta} = [\mathbf{\Theta}_\rho, \mathbf{\Theta}_\varphi, \mathbf{\Theta}_\mu, \mathbf{\Theta}_\gamma]$, with

$$\mathbf{\Theta}_\varphi = [\theta_\varphi^{(0)}, \theta_\varphi^{(1)}, \dots, \theta_\varphi^{(|\mathcal{W}_\varphi|-1)}], \quad \mathbf{\Theta}_\mu = [\theta_\mu^{(0)}, \theta_\mu^{(1)}, \dots, \theta_\mu^{(|\mathcal{W}_\mu|-1)}], \quad \mathbf{\Theta}_\gamma = [\theta_\gamma^{(0)}, \theta_\gamma^{(1)}, \dots, \theta_\gamma^{(|\mathcal{W}_\gamma|-1)}], \tag{4.10}$$

as the vectors of the model's feature extractor weights, multimodal representation weights, and multimodal fusion weights, respectively. We solve Equation (4.9) by optimizing the loss L:

$$L = L_{rec}(\mathbf{\Theta}, \hat{R}, R) + \alpha L_{reg}(\mathbf{\Theta}) \tag{4.11}$$

where R is the ground-truth value corresponding to the predicted output $\hat{R}$, and α is a model hyper-parameter to weight the *regularization* component of the loss function (i.e., L_{reg}). Algorithm 1 provides a general overview of the overall multimodal schema we presented.

Algorithm 1: Multimodal schema for multimedia recommendation

```
Input: Set of available modalities $\mathcal{M}$; set of multimodal input data $\mathcal{X}$ and
  admissible modalities $\mathcal{M}_x$, $\forall x \in \mathcal{X}$.
Output: Trained model's weights $\hat{\Theta}$.
Initialize all model's trainable weights $\boldsymbol{\Theta}$.
repeat
    extract features according to Equation (4.2)
    if Joint representation then
        get joint representation according to Equation (4.3)
        get model's prediction according to Equation (4.6)
    else if Coordinate representation then
        get coordinate multimodal representations according to Equation (4.4)
        if Early fusion then
            get multimodal representation according to Equation (4.5)
            get model's prediction according to Equation (4.6)
        else if Late fusion then
            get predictions for each modality according to Equation (4.7)
            get model's prediction according to Equation (4.8)
    for $\hat{\theta} \in \hat{\Theta}$ do
        update $\hat{\theta}$ according to Equation (4.9), by optimizing the loss function $L$
          in Equation (4.11)
    end
until convergence;
Return $\hat{\Theta}$.
```

4.4 Conceptual validation of the schema

After having described the formalism behind our proposed multimodal schema for multimedia recommendation, we devote the current section to the validation of the schema by conceptually applying it to four state-of-the-art approaches from the literature (Table 4.2). To demonstrate how our formal solution is designed to generally work with multiple recommendation scenarios involving multimedia user-item data interactions, we choose the proposed examples spanning a wide range of tasks, namely micro-video recommendation [339], food recommendation [321], outfit fashion compatibility [120], and artist/song recommendation [236].

4.4.1 Case 1: micro-video recommendation

Wei et al. [339] build a bipartite user-item graph for micro-video personalized recommendation. The idea behind the approach is to exploit high-order users-items relations leveraging the multimodal nature of recommended items (i.e., micro-videos), to which users may experience different attitudes. The authors adopt a graph convolutional network [158] to refine user and item embeddings (conditioned on the graph topology).

Multimodal input data

Micro-videos (the items) are described via three modalities, namely: *visual* (i.e., frames), *textual* (i.e., user-generated captions and descriptions) and *audio* (i.e., the audio track, that is not always available). It is worth pointing out that also users are described through three embeddings representing how each item modality might influence them differently. Nevertheless, they cannot be formally considered as multimodal input data (we do not report any information about the multimodal input data and feature extraction columns in the table).

Feature extraction

Visual features are extracted through a pretrained ResNet50 [122] only from key video frames. Textual features are derived from Sentence2Vector [24]. Audio features are extracted using a pre-trained VGGish [130].

Multimodal representation

The framework leverages three versions of the same bipartite user-item graph (i.e., one for each micro-video modality). The graph convolutional layer first aggregates the neighborhood features of the ego node and then combines the result of such aggregation with the collaborative embedding and the multimodal representation from the previous iteration. Given the formalism introduced above, we might say this approach goes under the definition of *Coordinate* representation. The model adopts a linear projection for each modality to map the input into a modality-specific latent space, both in the aggregation and combination steps. No explicit constraints are introduced.

Multimodal fusion

The adoption of a multimodal coordinate representation requires a modality fusion phase. This is performed through element-wise addition among modalities for users and items. As this occurs before feeding them into the inference function, we categorize it as *Early* fusion.

Inference and loss function

As in several collaborative filtering approaches, the inference is performed through the inner product between the multimodal representations of users and items. Regarding the loss function, the authors use the broadly-adopted BPR [258] optimization framework, maximizing the distance between predicted ratings for positive items (i.e., the ones interacted by users) and negative items (i.e., the ones not already interacted by users).

4.4.2 Case 2: food recommendation

Wang et al. [321] introduce a tripartite framework for food recommendation whose pipeline involves the retrieval of recipes according to user-generated videos, the profiling of users based upon their social media interactions, and the final health-aware food recommendation of recipes.

Multimodal input data

On the user side, it should be noticed that the input data does not properly follow the above definition we provide about multimodality, as users are profiled only according to the textual description of their generated tweets. However, we maintain the importance of this example since it represents one of the few approaches in the literature that proposes to model users through a multimodality-like solution. On the other side, items' description is multimodal because it integrates frames of user-generated videos to retrieve recipes from (i.e., *visual* modality) and descriptions of recipe ingredients (i.e., *textual* modality).

Feature extraction

Users' tweets are the input to what the authors define as a word-class interaction-based recurrent convolutional network (WIRCNN), which involves a recurrent neural network (RNN) and a convolutional neural network (CNN) to classify user tags. As for items, sampled video frames are encoded through a pre-trained VGGNet-19 [284], while textual recipe ingredients are processed via TextCNN [156].

Multimodal representation

Given that the user profile is not multimodal *per se*, we do not recognize any multimodal representation stage. On the item side, the multimodal representation is *Coordinate*, but no particular operation is performed on the extracted multimodal features.

Table 4.2 Four literature examples of multimodal frameworks for multimedia recommendation. For each work, we report the performed task, the considered modalities for each input to the system (e.g., user and item), the feature extraction and multimodal representation strategies, the multimodal fusion, and the adopted inference/loss functions.

	Paper	**Input**	**Modalities**	**Multimodal Input Data**	**Feature Extraction**	**Multimodal Representation**	**Multimodal Fusion**	**Inference and Loss**
Micro-video recommendation	Wei et al. [339]	User	Visual			Coordinate, aggregation of node's neighborhood and combination with the ego node. Projection in both aggregation and combination.	Early, modalities are combined through addition.	Inference via inner-product between final user and item embeddings. BPR is the optimization function.
			Textual					
			Audio					
		Item	Visual	Video frames	ResNet50			
			Textual	Captions and descriptions	Sentence2Vector			
			Audio	Audio track	VGGish			
Food recommendation	Wang et al. [321]	User	Textual	Users' tweets	Bi-RNN & CNN			Score prediction with MLP for user tags. Loss is cross-entropy. User embedding is later adopted for recommendation.
		Item	Visual	Video Frames	VGGNet-19	Coordinate, no particular operation performed.	Early, modalities are combined via concatenation.	User-item score prediction with MLP and cross-entropy.
			Textual	Recipe ingredients	TextCNN			
Outfit fashion compatibility	Han et al. [120]	Item	Visual	Product images	InceptionV3	Joint, features are projected into a shared embedding space.		The visual modality is used for the inference.
			Textual	Descriptions	One-hot-encode			The textual modality is used for the contrastive component of the loss function.
Artist and song recommendation	Oramas et al. [236]	Item	Textual	Artist biography	Custom CNN	Coordinate, textual and audio features normalized and optionally fed into two separate MLPs.	Early, either normalized features are concatenated and fed into a one-layer MLP, or multimodal representations are connected to the one-layer MLP.	Inference via inner-product between user and final item embeddings. Cosine distance is the loss function.
			Audio	Audio spectrogram	Custom CNN			

Multimodal fusion

No modality fusion is run over user profiles. Regarding items, authors adopt an *Early* modality fusion by concatenating the visual and textual features.

Inference and loss function

User embeddings are learned for tag prediction, with a one-layer MLP used to predict scores and cross-entropy as a loss function. The final user embeddings are eventually exploited for the main task of food recommendation. Contrarily, item embeddings are directly adopted for the score prediction, run with a one-layer MLP trained on a binary cross-entropy loss.

4.4.3 Case 3: outfit fashion compatibility

Han et al. [120] propose a framework to recommend the next fashion item that matches a set of already chosen ones to produce a visually appealing outfit. The authors address the task by considering the items composing a fashionable outfit as a temporal sequence, so they leverage a bidirectional LSTM [116].

Multimodal input data

Recommendation is multimodal because the authors adopt both product images (i.e., *visual* modality) and text descriptions of the fashion items extracted from the product details (i.e., *textual* modality).

Feature extraction

The visual features of fashion items are extracted from a GoogleNet InceptionV3 [298] pretrained network (the TFE), whose dimensionality is 2048. As for the textual description, each word is a one-hot-encoded vector.

Multimodal representation

Visual and textual extracted features are projected into a unique latent space, whose dimensionality is 512. According to the earlier formalism, this approach follows a *Joint* multimodal representation. On the one hand, 2048-dimensional visual features are compressed into a 512-dimensional embedding through a fully connected neural network layer, which is trained end-to-end with the recommendation model. On the other hand, the textual features for each description are first projected into the 512-dimensional latent space through linear mapping (i.e., adopting a projection matrix, which is also trained end-to-end). Then, the authors adopt bag-of-words to obtain a unique representation for the description of each fashion item.

Inference and loss function

Only the visual modality is adopted as input for the recommendation inference. However, the textual modality is exploited jointly with the visual input to minimize the contrastive component of the loss function, for whom the cosine similarity measures the distance between visual and textual modalities in the shared latent space.

4.4.4 Case 4: artist and song recommendation

The approach introduced by Oramas et al. [236] deals with the task of music recommendation. Specifically, the authors propose to divide the problem into artist and song recommendations by learning their separate embeddings and leveraging the textual artist biography and audio spectrogram as inputs.

Multimodal input data

Multimodality is to be found in the item's description, which is based upon artist biography (i.e., *textual* modality) and audio spectrogram derived from songs (i.e., *audio* modality).

Feature extraction

Artist biographies are processed through the state-of-the-art CNN, which is re-trained using word2vec word embeddings pre-trained on the Google News dataset [221]. As for the song latent factors, a custom CNN with 256, 512, and 1024 convolutional filters is trained on the time axis, having as output the 4096-dense layer.

Multimodal representation

Before further processing the extracted multimodal features, both textual and audio features are normalized. Afterward, they optionally go through two separate MLPs (i.e., *Coordinate* representation).

Multimodal fusion

The authors explore two possibilities: if no MLP processing occurred in the multimodal representation stage, then normalized features are concatenated and fed into a one-layer MLP; otherwise, multimodal representations are connected to the one-layer MLP. They adopt an *Early* multimodal fusion in both cases.

Inference and loss function

Inference is run through the inner product between user and item final embeddings, while cosine distance is the chosen loss function as the learned latent embeddings are l2-normalized.

4.5 Technical challenges

This section aims to overview the main technical challenges we recognize in multimodal approaches for multimedia recommendation. Starting from the proposed schema we presented in the previous sections, we outline the evident (or even less evident) issues emerging from the literature.

4.5.1 Missing modalities in the input data

Describing data under the lens of multimodality may be a two-sided coin. From one perspective, multimodality helps enrich the informative content carried by the input, thus exploring data's multi-faceted nature to learn better-tailored user-item preference patterns [208]. On the other side, the need to provide descriptive content for every input modality may come at the expense of some missing modalities (e.g., a video dataset could integrate videos having no textual content, for example, subtitles or descriptions may be only sometimes available). Tackling the modality misalignment in the data is a recent and widely discussed challenge in other domains [169, 201, 378, 381], and requires ad-hoc techniques to provide equal representation of all involved modalities to fully exploit their informative richness [202, 294]. Nevertheless, to the best of our knowledge, the issue remains open in recommendation.

4.5.2 Pre-trained feature extractors

Deep learning models processing images, texts, or audio have been shown to enrich the informative content of items' profiles in several recommendation algorithms. In most solutions, such architectures are used as pre-trained blocks to extract high-level features from the input data, thus exploiting the capability of deep neural networks to transfer knowledge among different datasets and/or tasks. Despite the ease of adopting ready-to-use feature extraction networks, we seek to underline a conceptual limitation that, to the best of our knowledge, is only partially investigated in the literature. Indeed, pre-trained representations extracted through state-of-the-art deep learning models are not necessarily supposed to capture those semantic features, which will likely captivate users for their final decision-making process. As an example, the embedding feature extracted from a product image (e.g., a bag) through a pre-trained deep convolutional network (e.g., ResNet50) is carrying high-level informative content driven by the task of *image classification*, but this does not mean the same knowledge will be helpful to predict whether the product could be *recommended* to a user.

4.5.3 Modalities representation

The multimodal representation of the extracted input data is among the main stages in the multimodal schema we described since it establishes the relationships for the selected input modalities. Nonetheless, the literature is not generally aligned on its definition since most of the works usually refer to *Joint* representation and *Early* fusion interchangeably. We recognize this as a conceptual issue because the two stages (i.e., representation and fusion) should be considered separately. We maintain that the former stands for the initial step to set interconnections among early-extracted multimodal features, while the latter, despite dealing again with modalities relationships, involves features that have been further processed towards the task optimization (i.e., recommendation in our case), thus embodying different rationales and techniques. Furthermore, the related literature suggests two possible solutions to multimodal representation, either *Joint* or *Coordinate*, where the latter additionally requires the subsequent fusion step. However, each of the paradigms' advantages and whether they might depend on the task remain under investigation.

4.5.4 Multimodal-aware fusion and optimization

While multimodal representation builds on input modalities in the early stages of the schema, multimodal fusion accounts for multimodal features that have already been processed, with a specific focus on the last steps (i.e., inference and model optimization). Similarly to what was observed above, multimodal fusion may come in the form of *Early* or *Late* fusion. The significant difference between the two approaches lies in preserving or not modalities separation during the inference (i.e., *Late* and *Early*, respectively). The literature demonstrates the vast predominance of *Early* solutions, whereas several works quite often refer to *Late* fusion by mistaking it for *Early* fusion. Indeed, providing a precise definition for the two is of paramount importance because the two approaches may serve different purposes. The rationale of *Late* fusion is to keep the modalities separation explicit during the inference phase so that the contribution of each modality is observable up to the last operation. Moreover, the literature is not aligned on the operation to fuse modalities. Non-trainable fusion functions (e.g., element-wise addition) are usually the preferred direction given that it is more lightweight and easy to perform to trainable approaches (e.g., neural networks) which (on their side) may allow to better tailor user-item preference prediction.

4.6 Summary

In this chapter, we highlighted the importance of formalizing the multimedia recommendation task under the lens of multimodal deep learning. By recognizing how the main recommendation approaches in the related literature fall into some recurrent strategy patterns, we outlined a unified multimodal schema that, following the established multimodal deep learning pipeline, formalizes the core stages of multimedia recommendation as: (i) multimodal input data, (ii) multimodal feature processing, (iii) multimodal feature fusion, and (iv) the multimodal recommendation task. After that, we applied the proposed schema to four multimedia recommendation scenarios. The proposed formal schema along with its conceptual application gave the opportunity to highlight technical challenges in the field of multimedia recommendation leveraging multimodal information. While the next chapter is devoted to analyze some of these challenges (i.e., "pre-trained feature extractors", "modalities representation", and "multimodal-aware fusion and optimization") we are still in the process of studying the interesting issue of "missing modalities in the input data". In this respect, you may refer to the preliminary proposal of an approach (based upon Feature Propagation [263]) to tackle the issue in the last chapter of this thesis.

Chapter 5

Leveraging the visual modality in multimedia recommendation

The current chapter studies and proposes novel solutions addressing each of the highlighted technical issues still existing in multimedia recommendation leveraging multimodal information; note that each of them may be mapped to a specific stage of the multimodal pipeline formalized in the previous chapter. It is important to mention that while the provided formalization is to be intended in a general multimedia recommendation setting, this chapter specifically takes into account visually-aware recommender systems, namely, the recommendation approaches working with product images. After an initial presentation of two complementary frameworks, Ducho and V-Elliot, aimed to the extraction of multimodal features for recommendation and reproducibility of visually-aware recommender systems, respectively, we propose to use them to benchmark visually-aware recommendation models with varying pre-trained deep learning networks for the visual extraction. Results motivate the consideration of two specific scenarios and settings, namely, fashion recommendation and adversarial attacks/defenses against visually-aware recommender systems. In the former, we propose a novel recommendation approach which disentangles the users' preferences at the granularity of content and style of images depicting clothes; in the latter, we investigate how and to what extent the human customers on e-commerce platforms may perceive the adversarial attacks on product images.

5.1 Ducho: an extractor for multimodal features

Despite being the initial stage of any multimodal recommendation pipeline, the extraction of meaningful multimodal features is paramount in delivering high-quality recommendations [82]. However, the current practice of employing diverse multimodal extraction procedures in each recommendation framework poses limitations. Firstly, these diverse implementations hinder the interdependence across various multimodal recommendation frameworks, making their fair comparison difficult [206]. Secondly, despite the availability of numerous pre-trained deep learning models in popular open source libraries, the lack of shared interfaces for feature extraction across them represents a challenge for model designers.

To address these shortcomings, we propose **Ducho**, a unified framework designed to streamline the extraction of multimodal features for recommendation systems. By integrating widely-adopted deep learning libraries as backends such as TensorFlow, PyTorch, and Transformers, we establish a shared interface that empowers users to extract and process audio, visual, and textual features from both items and user-item interactions (see Table 5.1). This abstraction allows to leverage methods from each backend without being encumbered by the specific implementation that backend poses. A notable feature of our framework lays in its easily configurable extraction pipeline, which can be personalized using a YAML-based file. Users can specify the desired models, their respective backends, and models' parameters (e.g., extraction layer).

By looking at the related literature, the most similar application to Ducho is Cornac [269], a framework for multimodal-aware recommendation. For the sake of completeness, we report their main differences. Differently from Cornac, Ducho: (i) is specifically aimed to provide customizable multimodal feature extractions, being completely agnostic to the downstream recommender system that might exploit the extracted features, thus being easily applicable to any model; (ii) provides the user with the possibility to select the deep learning extraction model, its backend, and its output layer; (iii) introduces the audio modality to the modalities set.

To foster the adoption of Ducho, we also develop a public Docker image pre-equipped with a ready-to-use CUDA environment[1], and propose three demos to show Ducho's functionalities. The GitHub repository, which comes with all needed resources is available at: https://github.com/sisinflab/Ducho.

5.1.1 Architecture

Ducho's architecture is built upon three main modules, namely, **Dataset**, **Extractor**, and **Runner**, where the first two modules provide different implementations depending on the specific modality (i.e., audio, visual, textual) taken into account. We also remind

[1]https://hub.docker.com/r/sisinflabpoliba/ducho.

Table 5.1 An overview of all modalities, sources, and backends combinations available in Ducho.

Modalities	**Sources**		**Backends**		
	Items	Interactions	TensorFlow	PyTorch	Transformers
Audio	✓	✓		✓	✓
Visual	✓	✓	✓	✓	
Textual	✓	✓			✓

the **Configuration** one among the other auxiliary components. The architecture is designed to be highly modular, possibly integrating new modules or customizing the existing ones. In the following, we dive deep into each outlined module/component.

Dataset

The **Dataset** module manages the loading and processing of the input data provided by the user. Starting from a general shared schema for all available modalities, this module provides three separate implementations: **Audio**, **Visual**, and **Textual** Datasets. As a common approach in the literature, the Audio and Visual Datasets require the path to the folder from which image/audio files are loaded, while the Textual Dataset works through a tsv file mapping all the textual characteristics to the inputs.

Noteworthy, and differently from other existing solutions, Ducho may handle each modality in two fashions, depending on whether the specific modality is describing either the **items** (e.g., product descriptions) or the **interactions** among users and items (e.g., reviews [19]). Concretely, while items are mapped to their unique ids (extracted from the filename or the tsv file), interactions are mapped to the user-item pair (extracted from the tsv file) they refer to. Although the pre-processing and extraction phases do not change at items- and interactions-level (see later), we believe this schema may perfectly suit novel multimodal-aware recommender systems with modalities describing every type of input source (even **users**).

Another important task for the Dataset module is to handle the pre-processing stage of data input. Depending on the specific modality involved, Ducho offers the possibility to:

- **audio:** load the input audio by extracting the waveform and sample rate, and re-sample it according to the sample rate the pre-trained model was trained on;
- **visual:** convert input images into RGB and resize/normalize them to align with the pre-trained extraction model;

- **textual:** (optionally) clean the input texts to remove or modify noisy textual patterns such as punctuation and digits.

After the extraction phase (see later), the Dataset module is finally in charge of saving the generated multimodal features into **numpy** array format following the file naming scheme from the previous mapping.

Extractor

The **Extractor** module builds an extraction model from a pre-trained network and works on each loaded/pre-processed input sample to extract its multimodal features. In a similar manner to the Dataset module, the Extractor provides three different implementations for each modality, namely, the **Audio**, **Visual**, and **Textual** Extractors. Ducho exposes a wide range of pre-trained models from three main backends: TensorFlow, PyTorch, and Transformers. The following modality/backend combinations are currently available:

- **audio:** PyTorch (Torchaudio) and Transformers;
- **visual:** Tensorflow and PyTorch (Torchvision);
- **textual:** Transformers (and SentenceTransformers).

To perform the feature extraction, Ducho takes as input the (list of) extraction layers for any pre-trained model. Since each backend handles the extraction of hidden layers within a network differently, we follow the guidelines provided in the official documentations, assuming that the user will follow the same naming/indexing scheme of the layers and know the structure of the selected pre-trained model in advance. The interested reader may refer to the README[2] under the `config/` folder on GitHub for an exhaustive explanation on how to set the extraction layer in each setting of modality and backend.

Finally, for the textual case, the user can also specify the specific task the pre-trained model should be trained on (e.g., sentiment analysis), as each pre-trained network may come with different versions depending on the training strategy.

Runner

The **Runner** module is the orchestrator of Ducho, whose purpose is to instantiate, call, and manage all the described modules. With its API methods, this module can

[2]https://github.com/sisinflab/Ducho/blob/main/config/README.md.

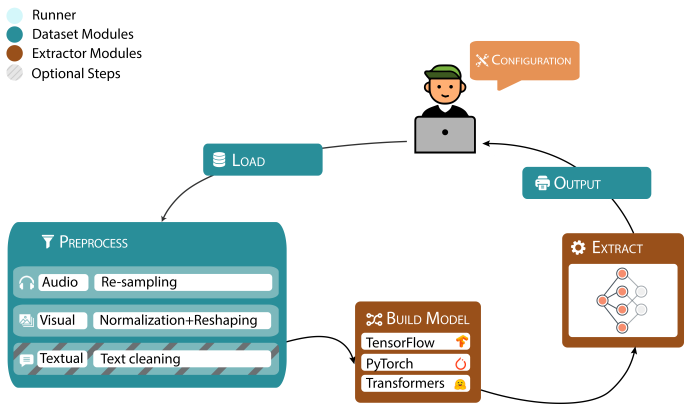


Fig. 5.1 Ducho's pipeline for multimodal feature extraction, managed by the Dataset, Extractor, and Runner modules.

trigger the complete extraction pipeline (see later) of one single modality or all the modalities involved simultaneously.

The Runner module is conveniently customized through an auxiliary **Configuration** component which stores and exposes all parameters to configure the extraction pipeline. Even if a default configuration is already made available for the user's sake, Ducho allows to override some (or all) its parameters through an external configuration file (in YAML format) and/or key-value pairs as input arguments if running the scripts from the command line. Once again, we suggest the readers refer to the README under the `config/` folder on GitHub to understand the general schema of the YAML configuration file.

5.1.2 Extraction pipeline

The overall multimodal extraction pipeline is represented in Figure 5.2. Through the Dataset module, the **load** and **preprocess** steps take place, assuming that the user is providing the input data and the YAML configuration file (overridable from command line) to customize the extraction. Then, the Extractor module is in charge of **building** the extraction model(s) by setting the backends and output layer(s). Finally, after the multimodal feature **extraction**, features are saved to the **output** path in numpy format (the Dataset module again controls this latter phase). As previously described, the whole process is orchestrated by the Runner module.

5.1.3 Ducho as Docker application

To fully exploit the GPU-speedup implemented in all backends we use for the multimodal feature extraction, one of the basic requirements is to setup a suitable development environment where the backends' versions are compatible with CUDA and, optionally, cuDNN. Generally, setting a workstation where all such libraries/tools are correctly aligned is challenging. To this end, we decide to dockerize Ducho by making it into a Docker image (available on Docker Hub[3]) with all packages already installed in a tested and safe virtualization environment on your physical machine.

Our Docker image is built from an NVIDIA-based image which comes with CUDA 11.8 and cuDNN 8 on Ubuntu 22.04, Python 3.8 and Pip, and our cloned repository having all Python packages already installed and ready to be used. A possible container instantiated from the image should specify the gpus to use from the host machine (this feature is currently available on Docker but it depends on the version of CUDA to be installed), and the volume you may want to use to save the framework's outputs.

Note that a generic container instantiated from our image would prompt the user to a shell environment where one could run custom multimodal feature extractions via the command line, and also create custom configuration files for the same purpose.

5.1.4 Demonstrations

This section proposes three use cases (i.e., demos) which show some of the main functionalities in Ducho and how to exploit them within a complete multimodal extraction pipeline. The guidelines and codes are accessible at this link[4] to run the demos (i) on your local machine, (ii) in a Docker container, and (iii) on Google Colab. Note that we specifically selected these demos as to replicate some real recommendation tasks involving multimodal features.

Demo 1: visual + textual items features

Fashion recommendation is probably one of the most popular task involving multimodal features to describe items. Generally, fashion products come with images (i.e., visual) and descriptions (i.e., textual) which may captivate the attention of the customer.
Input data. We use a small fashion dataset where each item has its own image and other metadata such as gender, category, colour, season, and product title. As for the visual modality, we save a subsample of 100 random images from the dataset in jpeg

[3]https://hub.docker.com/r/sisinflabpoliba/ducho.
[4]https://github.com/sisinflab/Ducho/tree/main/demos.

format; as for the textual modality, we produce for each of these items a description obtained as the combination of all the metadata fields from above, and store it into a tsv file where the first and second columns map item ids and descriptions, respectively. Note that, if no item column name is provided, Ducho selects, by default, the last column as the one holding the items' descriptions.

Extraction. In terms of extraction models, we adopt VGG19 and Xception for the product images, and Sentence-BERT pre-trained for semantic textual similarity for the descriptions. For each extraction model, we select the extraction layer, the pre-processing, and the library where the deep network should be retrieved from.

Output. Through the configuration file, we set Ducho to save the visual and textual embeddings to custom folders, where each embedding is a numpy array whose filename corresponds to the item name from the original input data. Additionally, Ducho keeps track of the log file in a dedicated folder within the project.

Demo 2: audio + textual items features

When it comes to recommending songs to users, audio and textual features may enhance the representation of each song, where the former are structured as a waveform, the latter as sentences referring, for instance, to the music genre of the song.

Input data. We use a small music genres dataset where each item comes with the binary representation of its waveform (we save it as wav audio track) and its music genre (we interpret it as textual song description and save it into a tsv file similarly to the previous demo). Given the heavy computational costs deep learning-based audio extractors require, we decide to select a small subset of the input songs (i.e., 10) just for the purpose of this demo.

Extraction. For the extraction of audio features we exploit Hybrid Demucs pre-trained for the task of music source separation. As for the textual extraction, we re-use the same deep neural model from the previous demo, since we are not interested in extracting other specific high-level features from music genres.

Output. Once again, we use the configuration file to specify the output folders for both the audio and textual embeddings. Please note that the extraction of audio features might take some time depending on the machine you are running Ducho on, as the deep audio extractor might require high computational resources to run.

Demo 3: textual items/interactions features

Online platforms usually allow customers to express reviews and comments about the products they have enjoyed to share their experience with other potentially-interested customers. In an e-commerce scenario, items may come with textual descriptions of the product characteristics (as seen in Demo 1). However, textual reviews of users commenting on those items may also be involved. Unlike most existing literature works, which usually refer to both sources of information as items' representations, we decide to conceptually distinguish between items- and interactions (i.e., user-item)-side representations for the former and the latter, respectively.

Input data. We adopt the widely-popular Amazon recommendation dataset where each user's purchase keeps track of metadata such as customer/product ids, the review text, the rating, and the purchase date. In a similar manner to the other demos, we retain only a small subset of the original dataset including 100 reviews and the corresponding product descriptions (obtained as the concatenation of their product title and category). Specifically, we save descriptions and reviews into separate tsv files where the former follow the same format as Demo 1 and Demo 2, while the latter maps user/item ids to review texts. Note that the number of products does not correspond to the number of user-item interactions as we only consider the set of unique interacted items. While Ducho extracts, by default, description/interaction texts from the last column of the tsv file, here we provide explicit column names to tell Ducho where to retrieve product descriptions and user reviews from the respective tsv files.

Extraction. While for the items' descriptions we use again the same sentences encoder as in Demo 1 and 2, we decide to extract textual features from users' reviews through a multilingual BERT-based model pre-trained on customers' reviews and specify the task of sentiment analysis for this model.

Output. Textual item features are saved to numpy arrays whose filenames are the item ids. Conversely, the textual interaction features are saved under the filename obtained from user and item ids to provide a unique pointer to each review.

5.2 Reproducing and evaluating visually-aware recommender systems

In some domains, such as fashion [137], food [95], or tourism [274], the visual appearance of a product image (e.g., piece of clothing or dish) is crucially important since it may affect user's final decision [124, 125]. Visual Recommender Systems (VRSs) integrate visual features of product images extracted through an image feature extractor (referred to as IFE, usually a CNN) into the recommendation pipeline to learn more tailored user profiles, overcoming issues such as data sparsity and cold-start [125].

The business of several online platforms is based on user-generated products and images (e.g., Pinterest, Amazon, Zalando, and Instagram). Consequently, Academia and Industry have channelled a considerable effort in designing novel approaches for visual recommendation [65, 142, 239]. Table 5.2 provides an outline of the most popular VRSs adopted as baselines in the recent literature, with some technical information on the input data type, the extraction layer and the training methodology for the IFE, and the official code link (if available).

Despite their adoption as baselines in several recent works (e.g., [16, 106, 344, 367, 374, 377]), to date nobody provided a unique framework implementing all these VRSs. Moreover, oftentimes, reproducibility is not even a feasible option since an official code is not always released (see "Code" in Table 5.2). Parra et al. [245] have recently proposed a tutorial on visual recommendation, presenting some (but not all) the above cited VRSs. Nevertheless, their work was not devoted to integrate the visual models into a complete framework for recommendation. Additionally, the copiousness of novel recommendation algorithms has generated confusion about choosing the correct baselines, the hyperparameter optimization, and the experimental evaluation to follow [267, 268]. Unreproducible evaluation and unfair comparisons [296] have recently arisen as a critical issue in the recommender systems community [80]. To this end, Anelli et al. [11] proposed Elliot, a framework for rigorous and reproducible recommender systems. The project is publicly available on GitHub, and provides several strategies for dataset loading, prefiltering, and splitting, along with hyperparameter optimization, recommendation models, and statistical hypothesis tests to build a reproducible experimental benchmark.

This second part of the chapter aims to provide a comprehensive demonstration of how to use Elliot for visual-based recommendation. Elliot for Visual recommendation (V-Elliot) implements all 6 VRSs from Table 5.2, with the possibility of leveraging: (i) a wide range of visual side information as input (e.g., the product image or its high-level visual feature extracted through a CNN-based IFE), (ii) a specific data input pipeline (implemented in `TensorFlow`) to efficiently handle memory-intensive streams of multidimensional data and inject them seamlessly into the recommendation flow, and (iii) an easy-to-use tool to train and test complex configurations of heterogeneous state-of-the-art (and custom) recommender systems by combining V-Elliot with the Elliot environment[5].

[5]The code is publicly available at: https://github.com/sisinflab/elliot.

Table 5.2 Most popular Visual Recommender Systems from the literature. For each work, we report its reference, publication year, adopted side information (i.e., either the image or the extracted visual feature of the item), the image feature extractor (with the chosen extraction layer and the training strategy), and link to the official code (if any). FC: fully-connected, FM: feature maps.

VRS	**Year**	**Side Info**		**Image Feature Extractor**				**Code**
				Extraction Layer		**Training**		
		Image	*Feature*	*FC*	*FM*	*Pretrained*	*End-to-End*	
VBPR [125]	2016		✓	✓		✓		×
DeepStyle [188]	2017		✓	✓		✓		×
DVBPR [151]	2017	✓		✓			✓	**[link]**
ACF [60]	2017		✓		✓	✓		**[link]**
VNPR [234]	2018		✓	✓		✓		×
AMR [302]	2020		✓	✓		✓		**[link]**

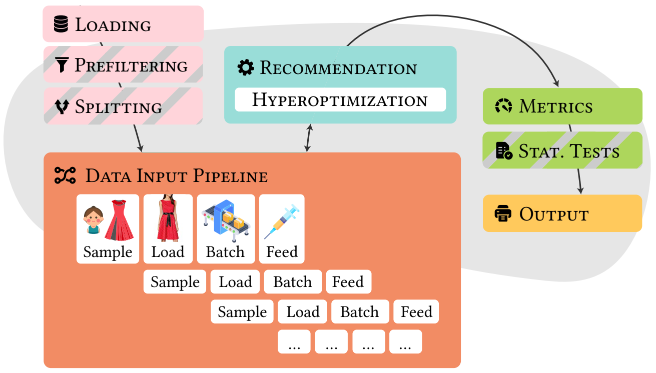


Fig. 5.2 Overview of V-Elliot. After the initial Loading (optionally complemented by Prefiltering and Splitting strategies), the Data Input Pipeline interacts with the Recommendation module to inject the visual data and train the model. The Metrics module evaluates the performance, whose values can be validated by statistical hypothesis tests. The Output module reports statistics and results.

5.2.1 V-Elliot: the visual recommendation framework

Elliot for Visual recommendation (V-Elliot) executes complex and reproducible experimental flows. As pointed out in Anelli et al. [11], the flexibility of the framework allows the user to design and run multiple possible settings through a concise configuration file, while seven modules are transparently loaded, each playing a specific functional role in the experimental flow. In addition to the already-existing modules (Figure 5.2), V-Elliot introduces a component to handle the loading and injection of visual side information (e.g., images and visual features) into the recommendation model.

The **Loading** module already supports various information sources (e.g., item features, semantic information [23], visual embeddings [125], and images [151]). As for the visual-based input data, the user can indicate the folder path where images (or features) are stored in separate files, which will be later injected *on-the-fly* into the framework when necessary. It is common knowledge that this strategy could alleviate the impact of memory-intensive experiments involving multidimensional visual data, which rarely can be pre-loaded into memory in advance. Users can also configure the settings for data pre-processing. In this respect, the **Prefiltering** module offers, among all, the possibility of applying the *filter-by-rating* and *k-core* strategies on the data, where the former removes user-item interactions whose preference score is smaller than a fixed (or data-based) threshold, and the latter filters out users, items, or both, with less than k interactions. Interestingly, the implementation of *k-core* algorithm also allows to retain cold users and items. Then, the **Splitting** module provides various temporal- and random-based splitting strategies, ranging from hold-out to cross-validation mechanisms. V-Elliot leverages a **Data Input Pipeline** to efficiently load visual-based input data and feed VRSs with it. The module is built upon the popular `TensorFlow` data input pipeline, which operates according to the producer/consumer paradigm, and consists of the following steps: (i) the next user-item interaction is sampled from the training set, (ii) visual data that has to be associated with the sample is loaded and (optionally) pre-processed, e.g., undergoing a normalization phase, (iii) samples are (optionally) grouped into batches, and (iv) the batches feed the recommendation algorithm. The **Recommendation** module interacts with the Data Input Pipeline, and integrates with an ever-growing set of state-of-the-art recommendation models seamlessly. To the best of our knowledge, V-Elliot is the framework providing the highest number of VRSs from the literature integrated into a complete system for recommendation (see again Table 5.2). Moreover, the simplicity of extending the set of available recommender systems through custom and external models, and an exhaustive number of hyper-parameter tuning strategies considerably ease the prototyping phase. The training procedure is assisted by the **Metrics** module that evaluates the model performance (with metrics ranging from accuracy to beyond-accuracy ones) and drives the selection of the best hyper-parameters configuration. Furthermore, the V-Elliot memory-optimized version of the visual-based Data Input Pipeline is also exploited to speed up the evaluation process. The evaluation phase may be further refined by computing two statistical hypothesis tests, i.e., *Wilcoxon* and *Paired t-test*, using the **Statistical Tests** module. Finally, V-Elliot collects the results through the **Output** module, which stores detailed performance tables, whereas model weights and recommendation lists may be saved for the sake of reproducibility, further analysis, and future experiments.

5.2.2 Execution of an experimental flow

Setting

To encourage researchers to try V-Elliot, we show the experiments run on two fashion datasets (i.e., `Amazon Baby` and `Amazon Boys & Girls` [124, 218]) filtered through the 5-core technique as suggested in He et al. [124, 125]. The final statistics are: 606 users, 1761 items, and 3882 interactions for `Amazon Baby`, and 1425 users, 5019 items, and 9213 interactions for `Amazon Boys & Girls`. For each item image, we have extracted high-level visual features with a pre-trained ResNet50 [122], following the findings shown in [82]. We split the data adopting the temporal leave-one-out protocol. To tune the hyper-parameters on the validation set, we performed a grid search using HR@100 as the validation metric. Table 5.3 presents the accuracy and beyond-accuracy metric values measured on the top-100 recommendation lists for each best model. The Elliot configuration files are reported in Table 5.3.

Results

Table 5.3 shows that ACF is the most accurate model on `Amazon Baby`, providing also the most novel recommendation lists (i.e., EFD and EPC) and being the second-to-best regarding diversity and coverage (i.e., Gini, SE, and iCov). Interestingly, DeepStyle settles as one of the most accurate models on `Amazon Boys & Girls`.

However, ACF still reaches remarkable accuracy results (it is the third-best recommender), confirming the performance observed on `Amazon Baby`. It is worth mentioning that the proposed analysis could be easily extended to wider search spaces, more metrics (e.g., bias measures), and additional (non-visual) recommender models (e.g., deep neural collaborative models), to eventually build an exhaustive evaluation workflow for recommendation.

5.2.3 The impact of pre-trained feature extractors

Given the representational power of convolutional neural networks (CNNs) in capturing characteristics and semantics of the images in supervised learning tasks, such as image classification, state-of-the-art VRSs often exploit pretrained CNNs to implement the Image Feature Extractor (IFE) component of a VRS, as shown in Figure 5.5. This approach allows VRSs to exploit: (i) the high-level visual representational power of CNNs, and (ii) their ability to generalize on datasets different from the ones they were trained on, e.g., ImageNet [88]. Despite their success, there is a lack of homogeneity in

Table 5.3 Measured accuracy and beyond-accuracy metrics for the tested Visual Recommender Systems and datasets on top-100 recommendation lists. Best values are reported in **bold**, while the second-best are underlined.

Model	Accuracy				Beyond-Accuracy				
	HR	nDCG	Precision	MAP	EFD	EPC	Gini	SE	iCov
`Amazon Baby` - [Elliot Configuration File: demo_amazon_baby.yml]									
VBPR	0.0743	0.0160	0.0007	0.0008	0.0075	0.0008	0.5730	10.0093	1386
DVBPR	0.0413	0.0082	0.0004	0.0004	0.0039	0.0004	0.1421	7.7011	370
ACF	**0.1221**	**0.0384**	**0.0012**	**0.0022**	**0.0169**	**0.0018**	0.6711	10.1862	1392
DeepStyle	0.0561	0.0117	0.0006	0.0005	0.0055	0.0006	**0.7490**	**10.2992**	**1393**
VNPR	0.0479	0.0112	0.0005	0.0006	0.0052	0.0005	0.2058	8.5920	965
AMR	0.0858	0.0192	0.0009	0.0010	0.0092	0.0009	0.5752	10.0083	1389
`Amazon Boys & Girls` - [Elliot Configuration File: demo_amazon_boys_girls0.yml]									
VBPR	0.0295	0.0068	0.0003	0.0004	0.0036	0.0003	0.4141	11.0699	3687
DVBPR	0.0309	0.0082	0.0003	0.0005	0.0038	0.0004	0.4692	11.2376	3842
ACF	0.0351	0.0075	0.0004	0.0003	0.0037	0.0004	0.0257	6.7975	120
DeepStyle	**0.0653**	**0.0210**	**0.0007**	**0.0013**	**0.0099**	**0.0010**	0.2886	10.5499	3176
VNPR	0.0260	0.0053	0.0003	0.0003	0.0029	0.0003	**0.6421**	**11.6361**	**3925**
AMR	0.0365	0.0094	0.0004	0.0005	0.0047	0.0004	0.5349	11.4075	3902

the selection of the pretrained networks in the literature, which usually happens to be a fixed choice. For instance, Hou et al. [133] propose an explainable fashion recommender system leveraging textual attributes, regions of item images, and a global visual profile of images extracted through AlexNet [164] , then Chen et al. [66] use VGG19 [284] to implement an explainable fashion recommender systems based upon image regions and user reviews, and finally Chen et al. [59] exploit ResNet50 [122] to generate a high-level description of recipe images which, along with textual descriptions, addresses the task of cross-modal recipe retrieval.

In this third part of the chapter, we aim at studying the impact of the three most popular pretrained CNN classes, namely AlexNet, VGG19, and ResNet50, used widely in the prior literature on a suite of competitive VRSs, contemplating four models, i.e., VBPR [125], DeepStyle [188], ACF [60], and VNPR [234]. The combinations of these CNNs and VRSs constitute the state-of-the-art for visual recommender models. Our contributions are two-fold: we evaluate to what extent different CNN architectural styles affect recommendation in terms of: (i) accuracy and beyond-accuracy metrics, and (ii) visual diversity of recommended items with respect to the ones previously consumed by each user.

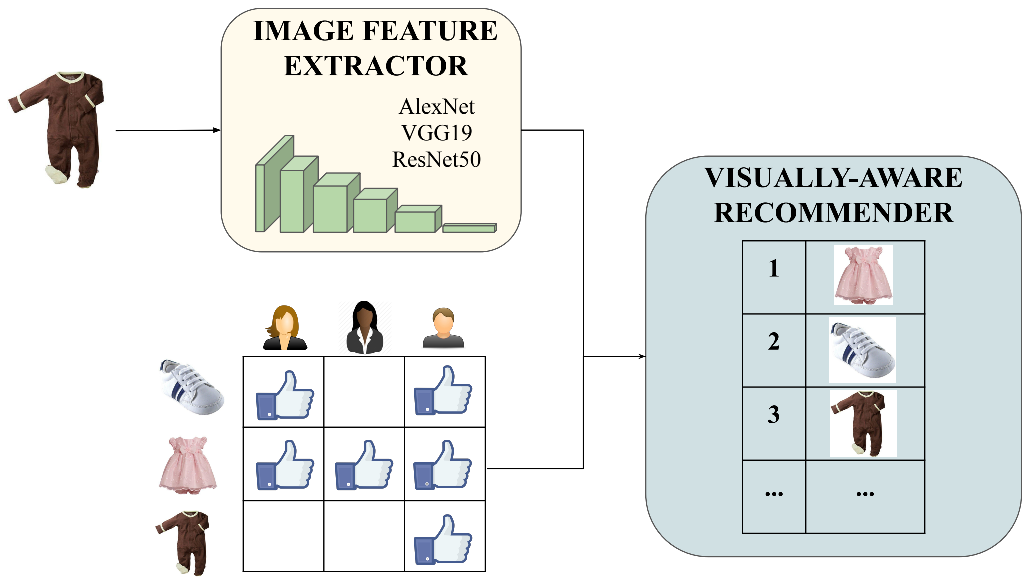


Fig. 5.3 A Visually-Aware Recommender System (VRS).

Visual recommendation problem

A visual recommendation problem (VRP) tailors recommendation problem to the cases where item images are available, e.g., fashion and food recommendation. Let $\mathcal{X}$ be the set of item images. We aim at finding the image feature extraction function f to obtain the visual features of each image $f(\mathbf{x}_i) = \boldsymbol{\varphi}_i$, with $\mathbf{x}_i \in \mathcal{X}$, enhancing, or even replacing, the recommendation-specific item representation. When pretrained CNNs are utilized as Image Feature Extractors (IFEs), it is common to extract the features on the layer activations, either convolutional or fully-connected.

Related work

Several works verified performance enhancements when integrating item visual features [74, 124, 125, 283]. The vast majority of them use high-level features extracted from CNNs, e.g., [124, 125], that could be either pretrained on a general-purpose dataset, e.g., ImageNet [88], or trained jointly with recommendation task, e.g., DVBPR [151]. As for the first category, VBPR [125] is the leading solution including visual features extracted from a pre-trained AlexNet [164] to extend the BPR-MF score function [258]. A year later, Liu et at. [188] proposed DeepStyle, a VBPR-based technique that assigns higher importance to the image style at the expense of the image category. Similarly, Niu et al. presented VNPR [234], which concatenates the PCA-reduced representation of item images extracted through an AlexNet-like architecture [403] to their recommendation embeddings before feeding it into a neural-based recommender model. Then, Chen et al. [60] implemented ACF, which (differently from the previous approaches) adopts the feature maps extracted from a convolutional layer of a pretrained ResNet152 [122] to weight the different regions within users' positive item images through attention mechanisms. Chen et al. [66] designed an attention-based approach for explainable fashion recommendations by exploiting a pretrained VGG19 [122].

While big efforts have been dedicated to building accurate VRSs, we noticed a lack in exploring how much the chosen pretrained CNN would impact on the recommendation performance. Indeed, we found that AlexNet, ResNet, and VGG are the most popular networks, i.e., at least 7 papers for the first [123–125, 133, 188, 218, 234], 6 for the second [16, 59, 60, 235, 302, 362], and 3 for the third one [66, 344, 346], but there are no exhaustive studies to verify their differences. In this respect, we aim to fill this gap by studying various configurations of state-of-the-art VRSs using standard pretrained CNNs, i.e., AlexNet, VGG19, and ResNet50.

Experiment Settings

Datasets. We investigate two fashion datasets, i.e., `Amazon Baby` and `Amazon Boys & Girls` [124, 218]. Both were filtered through the 5-core technique as suggested in [124, 125] to avoid cold-start users, thus resulting in the following statistics: the former counts 606 users, 1761 items, and 3882 registered interactions, while the latter covers 600 users and 2760 items, with 3910 ratings.

Image Feature Extractors. We study three IFEs: AlexNet, VGG19, and ResNet50. The first, AlexNet [164], is a 8-layer CNN, i.e., 5 convolutional and 3 fully-connected layers. This is one of the first architectures to introduce `ReLU` activation function [225] to address the saturation issue of the `tanh` function. The second, VGG19 [284], is one of the first *deep*-CNN, consisting of 19 layers, i.e., 16 convolutional and 3 fully-connected layers. All convolutions are built on a 3×3 kernel, and, like AlexNet, `ReLU` is the activation function. The last, ResNet50 [122], is the 50-deep CNN belonging to the ResNet family. It adopts residual blocks to tackle the training degradation problem observed in deep-CNNs. The ResNet family won the ILSVRC-2015 [191], outperforming their non-residual counterparts, e.g., VGG19.

Visual-based Recommender Models. We explore four VRSs: VBPR, DeepStyle, VNPR, and ACF. The first, Visual Bayesian Personalized Ranking (VBPR) [125], calculates the predicted rating for a user u and an item i as $\hat{R}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i + \boldsymbol{\theta}_u^\top \mathbf{W} \boldsymbol{\varphi}_i$, where $\boldsymbol{\theta}_u$ is the user's *visual* latent vector, $\boldsymbol{\varphi}_i$ is the item feature extracted from a

fully-connected layer, and $\mathbf{W}$ is an embedding matrix to project $\boldsymbol{\varphi}_i$ into $\boldsymbol{\theta}_u$'s space. DeepStyle [188], updates the VBPR score function by subtracting a $\mathbf{e}_u^\top \mathbf{c}_i$ term where $\mathbf{c}_i$ embodies the categorical information of i. Visual Neural Personalized Ranking (VNPR) [234], computes the (u, i) preference score with a MLP whose input is the concatenation of the element-wise product of $(\mathbf{e}_u, \mathbf{e}_i)$ and $(\mathbf{v}_u, \hat{\boldsymbol{\varphi}}_i)$, where the latter consists of the visual user profile and the PCA compression of $\boldsymbol{\varphi}_i$. Attentive Collaborative Filtering (ACF) [60], predicts the user's score of an unrated item using two attention networks to weigh its importance in the set of u-positive items and the regions within these images. The ACF feature is the feature map extracted from a conv layer.
Evaluation Metrics. We study accuracy and beyond-accuracy metrics evaluated on top-k recommendation lists. As for the *accuracy* measures, we adopt Recall@k, the fraction of recommended products in the top-k that hit test items and the AUC, a k-independent metric defined as the probability of ranking a positive item more than a random negative one. Then, the *beyond-accuracy* measures are iCov@k, the percentage of recommended items in the top-k lists and the EFD@k, a measure of the model capacity of suggesting relevant long-tail (unpopular) items [310]. All the above cited metrics range from 0 to 1, the closer to 1 the better.
Reproducibility. We split the datasets by adopting the temporal leave-one-out paradigm, i.e., for each user, the test and validation sets contain the last and second-to-last interactions. We apply a grid-search to tune the hyperparameters on the validation set. We release our code[6] implemented in Elliot [11].

Results and Discussion

This section evaluates the effects of varying the IFE on the top of the tested VRSs. All the metrics are computed for the top-100 recommendations. We will refer to each of them without the k term, e.g., iCov instead of iCov@100.
Analysis of Recommendation Results. Table 5.4 reports the accuracy and beyond-accuracy recommendation metrics. To begin with, it can be observed that VRSs built upon ResNet50 exhibit the best recommendation performance. Indeed, we notice that the VRS variants adopting visual features extracted from ResNet50 outperform the other IFE in 72% of the experimented cases. AlexNet settles as the second quality-level IFE, leaving VGG19 to the last place despite its widely-recognized ability to extract visual and stylistic content from images [146]. We may explain this, saying that deeper convolutional networks with residual blocks, such as ResNet50, produce more accurate recommendations thanks to their representational power.

[6]https://github.com/sisinflab/CNNs-in-VRSs

Additionally, we observe that the positive impact of ResNet50 on recommendation is uniformly not confirmed for ACF. In this setting, AlexNet is the pre-trained CNN that ensures the best accuracy performance in both the tested datasets. For instance, ACF using AlexNet features has a Recall equal to 0.0450, compared to the ResNet50 value of 0.0300. The reason for these outcomes could lie in the specific model characteristic. Indeed, differently from the other explored VRSs which take the output of a *fully-connected* layer as input, ACF leverages visual features extracted from a *convolutional* layer for the sake of the component-level attention mentioned in Section 5.2.3. As convolutional layers catch a lower-level representation of images compared to fully-connected ones, it entails that the different extraction layer is dramatically reducing the observed importance of IFE's depth in VRSs.

Furthermore, we evaluate the effects of varying the IFE on beyond-accuracy metrics, i.e., iCov and EFD. Similarly to the analysis of the accuracy-based results, both the beyond-accuracy measures reach the best values when ResNet50 is used as IFE. For example, considering the EFD measured for DeepStyle on `Amazon Baby`, the usage of ResNet50 produces the best metric value, i.e., 0.0271. In this setting, it is interesting to notice that only by changing the IFE from the original paper [188], i.e., AlexNet, we obtain an EFD improvement of +75%. This novel finding could be explained by the fact that the extracted features of deeper and complex CNNs, like ResNet50, allow learning more diverse users' preferences.

In summary, the results validate the hypothesis according to which the impact strength on VRSs can significantly vary based on the pretrained CNN employed. In fact, the deeper networks, such as ResNet50, seem to provide a much higher quality of recommendation in strong VRSs such as DeepStyle and VBPR. For average-quality VRSs, not a single CNN type outperforms the rest. Finally, we witness the same trend on beyond-accuracy metrics, such as item coverage and novelty, which directly measure the impact on users, platform owners, and third-party sellers in terms of economic gains and experience satisfaction [2, 166].

Analysis of Users Visual Profile. This section quantitatively and qualitatively evaluates to what extent each user's top-100 recommended items are visually similar, or dissimilar, to the list of positive ones. To address this analysis, we define the *visual diversity* (VisDiv@k) as the Euclidean distance between the visual features centroids extracted from both the positive and top-100 recommended items. Such distance is calculated after the application of the *t-SNE* algorithm to the feature embeddings to project them into a 2D latent space, which also come in handy for visualization purposes (see later).

Table 5.4 Recommendation results on top-100 lists.

Dataset	**VRS**	**IFE**	Recall	AUC	iCov	EFD
`Amazon Baby`	VBPR	AlexNet	0.1304	0.6308	0.9886	0.0142
		VGG19	0.1568	0.6344	0.9875	0.0162
		ResNet50	**0.2063**	**0.6475**	**0.9915**	**0.0246**
	DeepStyle	AlexNet	0.1337	0.6094	0.9994	0.0155
		VGG19	0.1683	0.6372	0.9960	0.0191
		ResNet50	**0.2195**	**0.6400**	**10.000**	**0.0271**
	ACF	AlexNet	**0.1271**	**0.5544**	**0.7910**	**0.0158**
		VGG19	0.1073	0.5477	0.7763	0.0132
		ResNet50	0.1023	0.5532	0.7791	0.0122
	VNPR	AlexNet	0.0561	0.5221	0.6303	0.0061
		VGG19	0.0891	0.5349	0.8001	0.0111
		ResNet50	**0.1221**	**0.5817**	**0.9733**	**0.0141**
`Amazon Boys & Girls`	VBPR	AlexNet	0.1033	0.6348	0.9808	0.0137
		VGG19	0.1133	0.6262	0.9681	0.0140
		ResNet50	**0.1250**	**0.6606**	**0.9837**	**0.0146**
	DeepStyle	AlexNet	0.0983	0.6160	0.9993	0.0114
		VGG19	0.1133	0.6307	**0.9996**	**0.0168**
		ResNet50	**0.1250**	**0.6402**	0.9957	0.0152
	ACF	AlexNet	**0.0450**	0.5120	**0.8043**	0.0047
		VGG19	0.0433	0.4955	0.7424	**0.0049**
		ResNet50	0.0300	**0.5235**	0.7518	0.0029
	VNPR	AlexNet	0.0317	0.5018	0.5319	0.0043
		VGG19	0.0417	0.5358	0.6272	0.0051
		ResNet50	**0.0800**	**0.5727**	**0.9667**	**0.0094**

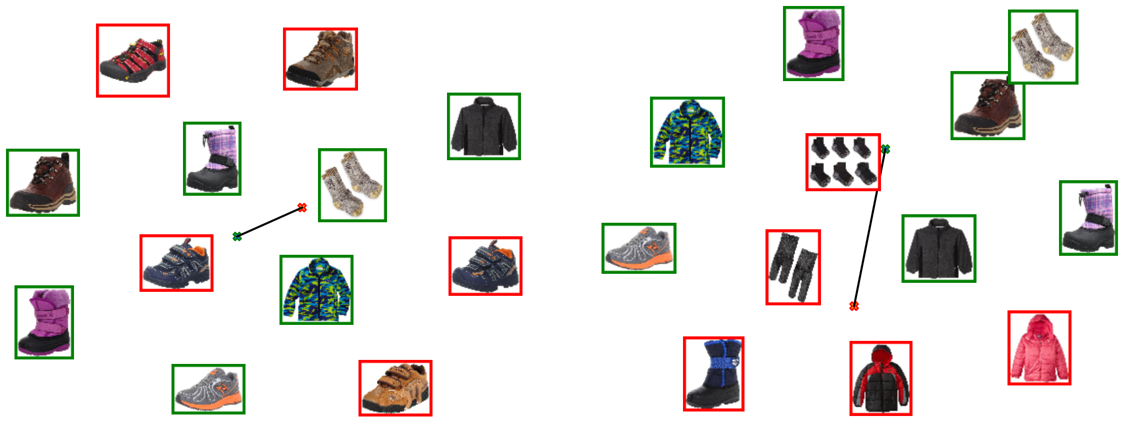
Table 5.5 reports the average VisDiv on all users. Investigating this quantitative metric, it can be observed that the settings with higher VisDiv are connected to the ones with the most *accurate* and *diverse* recommendation performance in Table 5.4. For instance, when comparing VBPR experiments varying the IFE, both visual and recommendation metrics reach the highest values when using ResNet50. To be specific, VisDiv, i.e., 17.05 and 20.67, Recall, i.e., 0.2063 and 0.1250, EFD, i.e., 0.0246 and 0.0146, on `Amazon Baby` and `Amazon Boys & Girls` respectively, confirm that a higher VisDiv value can be linked to better recommendation performance. Coherently, comparing the bold values of Table 5.4 and Table 5.5, it can be seen that VRSs using the IFE with ResNet50 produce the best performing and most visually-diverse recommendations.

To conclude, Figure 5.4 helps to inspect the visual differences of the positive and top-5 VBPR-based recommended items of a user sampled from `Amazon Boys & Girls` when the image features are extracted from AlexNet (Figure 5.4a) and ResNet50

Table 5.5 Average *visual diversity* (**VisDiv**) on top-100 lists.

Dataset	**VRS**	**IFE**		
		AlexNet	VGG19	ResNet50
`Amazon Baby`	VBPR	13.16	14.92	**17.05**
	DeepStyle	14.52	14.10	**16.64**
	ACF*	53.93	**59.93**	52.27
	VNPR	7.40	**20.75**	10.32
`Amazon Boys & Girls`	VBPR	10.16	15.62	**20.67**
	DeepStyle	12.32	14.27	**20.08**
	ACF*	58.46	**70.73**	48.31
	VNPR	11.96	8.98	**27.27**

* *Visual features have been flattened for t-SNE.*



(a) **AlexNet** (b) **ResNet50**

Fig. 5.4 Positive (green) and top-5 (red) item features in the latent space for (a) AlexNet and (b) ResNet50. The VisDiv@5 (the line connecting the two centroids) are 67.83 and 416.22 respectively.

(Figure 5.4b). It can be observed that, while the usage of AlexNet leads to the recommendation of items visually similar to the positive ones, i.e., all items are in the "trekking shoes" category as shown in Figure 5.4a, the application of ResNet50 makes recommendations more diverse, i.e., boots and socks in Figure 5.4b, and even with variable colour, e.g., the recommended jackets.

5.3 Content-style item representation for visually-aware recommendation

Recently, there have been a few attempts trying to uncover user's personalized visual attitude towards finer-grained item characteristics, e.g., [60, 66, 69, 133]. These solutions disentangle product images at (i) content-level, by adopting item metadata and/or reviews [69, 238], (ii) region-level, by pointing the user's interest towards parts of the image [66, 346] or video frames [60], and (iii) both content- and region-level [133]. Indeed, most of these approaches [60, 66, 133, 346] exploit attention mechanisms to weight the importance of the content or the region in driving the user's decisions.

Despite their superior performance, we recognize practical and conceptual limitations in adopting both content- and region-level item features, especially in the fashion domain. The former rely on additional side information (e.g., image tags or reviews), which could be not-easily and rarely accessible, as well as time-consuming to collect, while the latter ignore stylistic characteristics (e.g., color or texture) that can be impactful on the user's decision process [413].

Driven by these motivations, we propose a pipeline for visual recommendation, which involves a set of visual features, i.e., color, shape, and category of a fashion product, whose extraction is straightforward and always possible, describing items' content on a stylistic level. We use them as inputs to an attention- and neural-based visual recommender system, with the following purposes:

- We disentangle the visual item representations on the stylistic content level (i.e., color, shape, and category) by making the attention mechanisms weight the importance of each feature on the user's visual preference and making the neural architecture catch non-linearities in user/item interactions.
- We reach a reasonable compromise between accuracy and beyond-accuracy performance, which we further justify through an ablation study to investigate the importance of attention (in all its configurations) on the recommendation performance. Notice that no ablation is performed on the content-style input features, as we learn to weight their contribution through the end-to-end attention network training procedure.

Code and datasets to reproduce our model are available at: https://github.com/sisinflab/Content-Style-VRSs.

5.3.1 Method

In the following, we present our visual recommendation pipeline (Figure 5.5).

Let $\mathcal{S}$ be the set of content-style features to characterize item images. Even if we adopt $\mathcal{S} = \{\text{color}, \text{shape}, \text{category}\}$, for the sake of generality, we indicate with $\mathbf{f}_i^s \in \mathbb{R}^{1 \times v_s}$ the s-th content-style feature of item i. Since all $\mathbf{f}_i^s$ do not necessarily belong to the same latent space, we project them into a common latent space $\mathbb{R}^{1\times d}$, i.e., the same as the one of $\mathbf{e}_u$ and $\mathbf{e}_i$. Thus, for each $s \in \mathcal{S}$, we build an encoder function $enc_s : \mathbb{R}^{1\times v_s} \mapsto \mathbb{R}^{1\times d}$, and encode the s-th content-style feature of item i as:

$$\mathbf{e}_i^s = enc_s(\mathbf{f}_i^s) \tag{5.1}$$

where $\mathbf{e}_i^s \in \mathbb{R}^{1\times d}$, and enc_s is either trainable, e.g., a multi-layer perceptron (MLP), or handcrafted, e.g., principal-component analysis (PCA). We use an MLP-based encoder for the color feature, a CNN-based encoder for the shape, and PCA for the category.
Attention Network. We seek to produce recommendations conditioned on the visual preference of user u towards each content-style item characteristic. That is, the model is supposed to assign different importance weights to each encoded feature $\mathbf{e}_i^s$ based on the predicted user's visual preference ($\hat{r}_{u,i}$). Inspired by previous works [60, 66, 133, 346], we use attention. Let $ian(\cdot)$ be the function to aggregate the inputs to the attention network $\mathbf{e}_u$ and $\mathbf{e}_i^s$, e.g., element-wise multiplication. Given a user-item pair (u,i), the network produces an attention weight vector $\mathbf{a}_{u,i} = [a_{u,i}^0, a_{u,i}^1, \dots, a_{u,i}^{|\mathcal{S}|-1}] \in \mathbb{R}^{1\times|\mathcal{S}|}$, where $a_{u,i}^s$ is calculated as:

$$a_{u,i}^s = \boldsymbol{\omega}_2(\boldsymbol{\omega}_\mathbf{1} ian(\mathbf{e}_u, \mathbf{e}_i^s) + \mathbf{b}_1) + \mathbf{b}_2 = \boldsymbol{\omega}_2(\boldsymbol{\omega}_\mathbf{1}(\mathbf{e}_u \odot \mathbf{e}_i^s) + \mathbf{b}_1) + \mathbf{b}_2 \tag{5.2}$$

where $\odot$ is the Hadamard product (element-wise multiplication), while $\boldsymbol{\omega}_*$ and $\mathbf{b}_*$ are the matrices and biases for each attention layer, i.e., the network is implemented as a 2-layers MLP. Then, we normalize $\mathbf{a}_{u,i}$ through the temperature-smoothed $softmax$ function [131], so that $\sum_s a_{u,i}^s = 1$, getting the normalized weight vector $\boldsymbol{\alpha}_{u,i} = [\alpha_{u,i}^0, \alpha_{u,i}^1, \dots, \alpha_{u,i}^{|\mathcal{S}|-1}]$. We leverage the attention values to produce a unique and weighted stylistic representation for item i, conditioned on user u:

$$\mathbf{w}_i = \sum_{s\in\mathcal{S}} \alpha_{u,i}^s \mathbf{e}_i^s \tag{5.3}$$

Finally, let $oan(\cdot)$ be the function to aggregate the latent factor $\mathbf{q}_i$ and the output of the attention network $\mathbf{w}_i$ into a unique representation for item i, e.g., through addition. We calculate the final item representation $\mathbf{q}_i'$ as:

$$\mathbf{e}_i' = oan(\mathbf{e}_i, \mathbf{w}_i) = \mathbf{e}_i + \mathbf{w}_i \tag{5.4}$$

Neural Inference. To capture non-linearities in user/item interactions, we adopt an MLP to run the prediction. Let $concat(\cdot)$ be the concatenation function and $out(\cdot)$ be

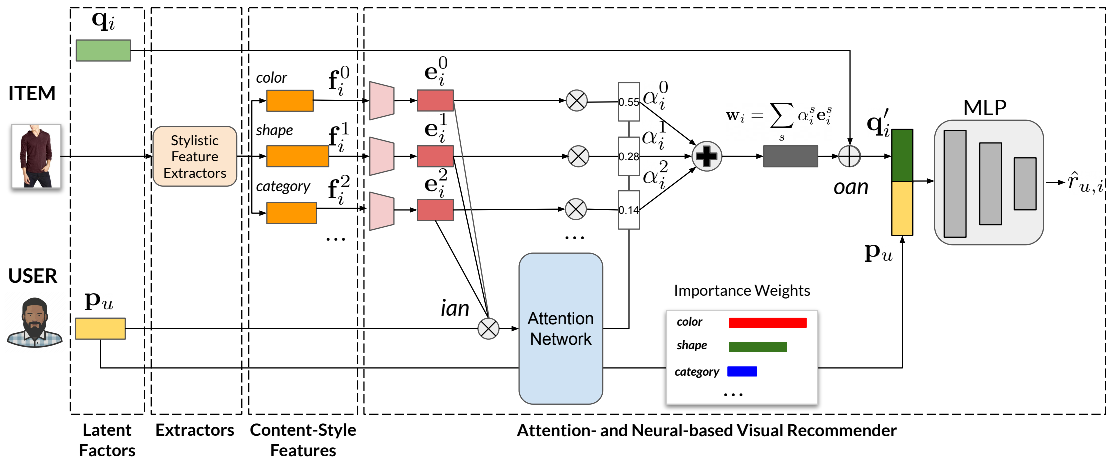


Fig. 5.5 Our proposed pipeline for visual recommendation, involving content-style item features, attention mechanisms, and a neural architecture.

a trainable MLP, we predict rating $\hat{r}_{u,i}$ for user u and item i as:

$$\hat{R}_{u,i} = out(concat(\mathbf{e}_u, \mathbf{e}_i')) \tag{5.5}$$

Objective Function and Training. We use Bayesian personalized ranking (BPR) [258]. Given a set of triples $\mathcal{T}$ (user u, positive item p, negative item n), we seek to optimize the following objective function:

$$\underset{\mathbf{\Theta}}{\operatorname{argmin}} \sum_{(u,p,n)\in\mathcal{T}} -\ln(sigmoid(\hat{R}_{u,p} - \hat{R}_{u,n})) + \lambda||\Theta||^2 \tag{5.6}$$

where $\mathbf{\Theta}$ and λ are the set of trainable weights and the regularization term, respectively. We build $\mathcal{T}$ from the training set by picking, for each randomly sampled (u,p) pair, a negative item n for u (i.e., not-interacted by u). Moreover, we adopt mini-batch Adam [157] as optimizing algorithm.

5.3.2 Experiments

Datasets. We use two popular categories from the Amazon dataset [124, 218], i.e., Boys & Girls and Men. After having downloaded the available item images, we filter out the items and the users with less than 5 interactions [124, 125]. Boys & Girls counts 1,425 users, 5,019 items, and 9,213 interactions (density is 0.00129), while Men counts 16,278 users, 31,750 items, and 113,106 interactions (density is 0.00022). In both cases, we have, on average, > 6 interactions per user.

Feature Extraction and Encoding. Since we address a fashion recommendation task, we extract color, shape/texture, and fashion category from item images [304, 413]. Unlike previous works, we leverage such features because they are easy to extract and always accessible and represent the content of item images at a stylistic level. We extract the **color** information through the 8-bin RGB color histogram, the **shape/texture** as done in [304], and the **fashion category** from a pretrained ResNet50 [59, 82, 102, 362], where "category" refers to the classification task on which the CNN is pretrained. As for the features encoding, we use a trainable MLP and CNN for color (a vector) and shape (an image), respectively. Conversely, following [234], we adopt PCA to compress the fashion category feature, also to level it out to the color and shape features that do not benefit from a pretrained feature extractor.

Baselines. We compare our approach with pure collaborative and visual-based approaches, i.e., BPRMF [258] and NeuMF [128] for the former, and VBPR [125], DeepStyle [188], DVBPR [151], ACF [60], and VNPR [234] for the latter.

Evaluation and Reproducibility. We put, for each user, the last interaction into the test set and the second-to-last into the validation one (i.e., temporal leave-one-out). Then, we measure the model accuracy with the hit ratio (HR@k, the validation metric) and the normalized discounted cumulative gain (nDCG@k) as performed in related works [60, 128, 389]. We also measure the fraction of items covered in the catalog (iCov@k), the expected free discovery (EFD@k) [310], and the diversity with the 1's complement of the Gini index (Gini@k) [118]. For the implementation, we used the framework Elliot [11, 12].

5.3.3 Results

What are the accuracy and beyond-accuracy recommendation performance? Table 5.6 reports the accuracy and beyond-accuracy metrics on top-20 recommendation lists. On Amazon Boys & Girls, our solution and DeepStyle are the best and second-best models on accuracy and beyond-accuracy measures, respectively (e.g., 0.03860 vs. 0.03719 for the HR). In addition, our approach outperforms all the other baselines on novelty and diversity, covering a broader fraction of the catalog (e.g., iCov $\simeq$ 90%). As for Amazon Men, the proposed approach is still consistently the most accurate model, even beating BPRMF, whose accuracy performance is superior to all other visual baselines. Considering that BPRMF covers only the 0.6% of the item catalog, it follows that its superior performance on accuracy comes from recommending the most popular items [39, 214, 411]. Given that, we maintain the competitiveness of our solution, being the best on the accuracy, but also covering about 29% of the item

Table 5.6 Accuracy and beyond-accuracy metrics on top-20 recommendation lists.

Model	HR	nDCG	iCov	EFD	Gini
		Amazon Boys & Girls — `configuration file`			
BPRMF	.01474	.00508	.68181	.00719	.28245
NeuMF	.02386	.00999	.00638	.01206	.00406
VBPR	.03018	.01287	.71030	.02049	.30532
DeepStyle	.03719	.01543	.85017	.02624	.44770
DVBPR	.00491	.00211	.00438	.00341	.00379
ACF	.01544	.00482	.70731	.00754	.40978
VNPR	.01053	.00429	.51584	.00739	.13664
Ours	**.03860**	**.01610**	**.89878**	**.02747**	**.49747**
		Amazon Men — `configuration file`			
BPRMF	.01947	.00713	.00605	.00982	.00982
NeuMF	.01333	.00444	.00076	.00633	.00060
VBPR	.01554	.00588	.59351	.01042	.17935
DeepStyle	.01634	.00654	**.84397**	**.01245**	**.33314**
DVBPR	.00123	.00036	.00088	.00069	.00065
ACF	.01548	.00729	.19380	.01147	.02956
VNPR	.00528	.00203	.59443	.00429	.16139
Ours	**.02021**	**.00750**	.28995	.01242	.06451

Table 5.7 Ablation study on different configurations of attention, *ian*, and *oan*.

Components		Boys & Girls		Men	
***ian*(·)**	***oan*(·)**	HR	iCov	HR	iCov
No Attention		.01263	.01136	.01462	.02208
Add	Add	.02316	.00757	.02083	.00076
Add	Mult	.02246	.00458	.00768	.00079
Concat	Add	.01404	.00518	**.02113**	.00076
Concat	Mult	.02456	.00458	.00891	.00085
Mult	*Add*	**.03860**	**.89878**	.02021	**.28995**
Mult	Mult	.02807	.00478	.01370	.01647

catalog and supporting the discovery of new products (e.g., EFD = 0.01242 is the second to best value). That is, the proposed method shows a competitive performance trade-off on accuracy and beyond-accuracy metrics.

How performance is affected by different configurations of attention, *ian*, and *oan*?

Following [66, 133], we feed the attention network by exploring three aggregations for the inputs of the attention network (*ian*), i.e., element-wise multiplication/addition and concatenation, and two aggregations for the output of the attention network (*oan*), i.e., element-wise addition/multiplication. Table 5.7 reports the HR, i.e., the validation metric, and the iCov, i.e., a beyond-accuracy metric. No ablation study is run on the content-style features, as their relative influence on recommendation is learned during the training. First, we observe that attention mechanisms, i.e., all rows but *No Attention*, lead to better-tailored recommendations. Second, despite the {Concat, Add} choice reaches the highest accuracy on Men, the {Mult, Add} combination we used is the most competitive on both accuracy and beyond-accuracy metrics.

5.4 Adversarial attacks and defenses in visually-aware recommendation

The literature has shown that deep neural networks (DNNs) are vulnerable to adversarial examples [36, 299] minimal-corrupted images crafted to fool the network. Szegedy et al. [299] formalized the adversarial generation problem by solving a box-constrained L-BFGS. Goodfellow et al. [115] used the sign of the gradient of the loss function to perturb the images in the Fast Gradient Sign Method (FGSM). Madry et al. [204] adapted FGSM and Basic Iterative Method [114] to *iteratively* update the perturbation and get stronger adversarial samples. Carlini et al. [50] (C & W) boosted the Szegedy et al. [299] strategy to craft powerful samples able to deceiving state-of-the-art adversarial detector [49]. However, the Adversarial Training, proposed by Goodfellow et al. [115], has demonstrated substantial DNN's protection when adversarial samples are injected into the training data at a long-time training cost. This issue has been recently addressed by Shafahi et al. [275] with the proposal of the $3-30$ times faster Free Adversarial Training.

Consequently, adversarially-perturbed product images have been also shown to fool the DNNs used in visually-aware recommender systems (VRs) to extract the visual features [84]. Tang et al. [302] tested the accuracy degradation when VBPR is trained on noisy images (integrity attack), while Noia et al. [235] demonstrated the adversary's capability to increase (or decrease) the recommendability of a category of products (integrity attack) even on the *adversarial regularized* [127] version of VBPR, namely AMR [302].

In this last part of the chapter, we investigate the efficacy of defensive mechanisms [115, 275] against powerful attacks [48, 115, 204] when the adversary wants to alter the recommendation lists of a VRS by *poisoning* the training data by inserting adversarial product images, e.g., one perturbs images of low popular products so that they are misclassified as popular ones. Furthermore, we provide a visual-oriented evaluation of adversarial images through offline *visual metrics trying to mimicking human evaluation* to verify to what extent users might become aware of such subtle data poisoning in the received recommendations (Figure 5.6).

The main contributions are twofold: (1) we verify the inefficacy of state-of-the-art adversarial training procedure in defending the DNNs used in VRS from *adversarially-poisoned training product images*; (2) we evaluate the human-perceptibility with offline measures. Source code, data, and experimental parameters are available at: https://github.com/sisinflab/Perceptual-Rec-Mutation-of-Adv-VRs.

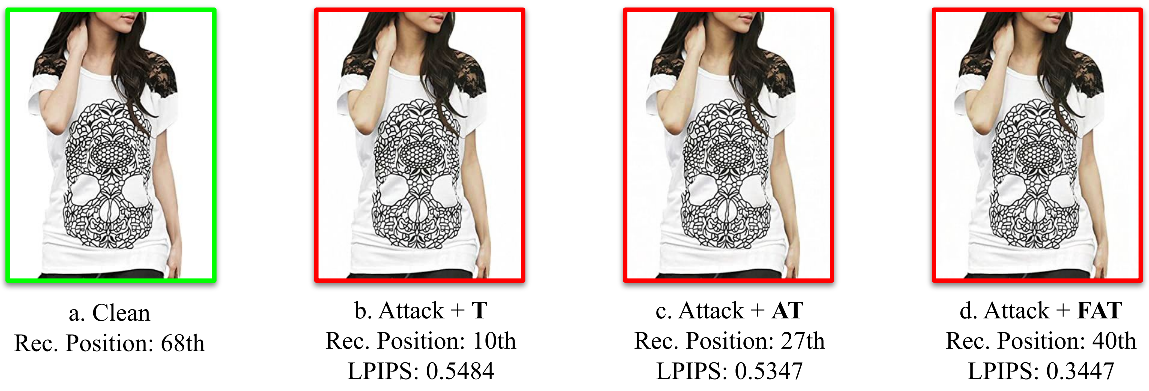


Fig. 5.6 (a) is the image of a *low-recommended* product. (b, c, d) are the perturbed versions with PGD ($\epsilon = 8$) applied against DNNs without defense (T), or with the Adversarial Training (AT) and Free AT (FAT). The attacks have pushed the product towards *higher* ranking positions without *visually-perceptible* artifacts.

5.4.1 The threat model

The dependence of a VRS from visual features extracted from pre-trained DNNs has been exploited by adversaries to poison the training data with the insertion of adversarial samples [195, 235, 302]. To generate the **targeted** adversarial attack the optimization problem formulation is:

$$\max_{\boldsymbol{\delta}_i : \|\boldsymbol{\delta}_i\|_p \leq \epsilon} \mathcal{L}_F(\mathbf{x}_i + \boldsymbol{\delta}_i, y_i) \text{ s.t. } y_i = m \tag{5.7}$$

where F is a DNN, $\mathcal{L}_F$ is the cost function of F, $\boldsymbol{\delta}_i$ is the ϵ-bounded perturbation of $\mathbf{x}_i$ that will make the product image be misclassified by F as the (more popular) product category m, and $\|\cdot\|_p$ is the L_p norm. For instance, the adversary can poison the data adding a perturbed image of "*Jersey, T-shirt*" misclassified as "*Brassiere*" (Fig. 5.6) causing a variation in the VRS since $\mathbf{f}_i$ will be extracted from $\mathbf{x}_i^{adv} = \mathbf{x}_i + \boldsymbol{\delta}_i$.

Recently, studies on the robustification of DNNs have shown the adversarial training by Goodfellow et al. [115] is one of the most prominent defense technique. After the definition of the adversary threat model (i.e., the attack strategy), the adversarial minimax formulation is:

$$\min_{\widetilde{\boldsymbol{\theta}}} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{I}} \max_{\boldsymbol{\delta}_i : \|\boldsymbol{\delta}_i\|_p \leq \epsilon} \mathcal{L}_F(\mathbf{x}_i + \boldsymbol{\delta}_i,\ y_i) \tag{5.8}$$

where $\widetilde{\boldsymbol{\theta}}$ represents the model parameters of the robustified network ($\widetilde{F}$).

Let $\widetilde{\mathbf{f}}_i$ the visual features of the image $\mathbf{x}_i$ associated to a product image extracted from $\widetilde{F}$. In this work, we want to verify if the application of adversarial training methods can limit poisoning attacks against VRSs [235, 302] since each user-item score prediction $\hat{R}_{ui}$ depends on $\widetilde{\mathbf{f}}_i$. Furthermore, we want to investigate whether the usage of adversarial trained DNNs will make the adversarial perturbation evident to such an extent that it makes the perturbed samples identifiable via a human evaluation.

5.4.2 Experiments

Setup

The experiments are conducted on two fashion datasets, i.e., `Amazon Women` and `Amazon Men` made publicly available by He et al. [124]. They come with both users' ratings and product pictures uploaded by the platform owner and third-party sellers (say, the possible adversaries). `Amazon Women` counts 16668 users, 2981 items, and 54473 ratings, while `Amazon Men` counts 24379, 7371, and 89020. We split the data following the time-aware leave-one-out protocol [127].

To empirically study the efficacy of defenses and evaluate the visual appearance of adversarial samples, we tested two VRS: VBPR by He et al. [125], and AMR by Tang et al. [302], a VBPR extension that includes the adversarial regularizer of visual features proposed by He et al. [127]. The complete set of experimental parameters is reported in the GitHub repository.

Evaluation of Recommendation Performance

Table 5.8 shows the recommendation variation before and after the attacks. We evaluate the variation of recommendation with the Category Hit Ratio [235], that measures the average number of a (pushed) category of items in the top-K recommendation lists. In particular, results in Table 5.8 are measured on the following source-target combinations: "*Sandal*"-"*Running Shoe*" for `Amazon Men`, while "*Jersey, T-shirt*"-"*Brassiere*" for `Amazon Women`, where the adversary tries to push a source category by perturbing the product picture to be classified as a target class, e.g., the class of a very popular category.

Analyzing VBPR outcomes, PGD attack shows the highest variation of Category Hit Ratio @ 20 in the defense-free experiments. For instance, PGD ($\epsilon = 8$) increases by more than 2.3 times the Category Hit Ratio @ 20 of the source category in the <`Amazon Women`, VBPR, Traditional> setting. The same trend is not true for the defense contexts. C&W attacks have increased the Category Hit Ratio @ 20 by 71.09%, while PGD ($\epsilon = 8$) by 69.35%. Furthermore, Table 5.8 confirms that the adversarial

Table 5.8 Category Hit Ratio @ 20 results on `Amazon Women` and `Amazon Men`. We mark in **bold** the most effective attacks.

Model	Attack	`Amazon Women`			`Amazon Men`		
		T	**AT**	**FAT**	**T**	**AT**	**FAT**
VBPR	No-Attack	*0.4377*	*0.5108*	*0.3417*	*0.6352*	*0.3028*	*0.3702*
	FGSM ($\epsilon = 4$)	0.3860	0.6032	0.6088	0.5665	0.6029	0.5688
	FGSM ($\epsilon = 8$)	0.4057	0.6186	**0.6313**	0.6052	0.5879	0.5596
	PGD ($\epsilon = 4$)	0.4377	0.6309	0.6263	1.0936	0.6211	0.5778
	PGD ($\epsilon = 8$)	**1.4462**	**0.6413**	0.6139	**1.5736**	0.6247	0.6141
	C&W	0.4147	0.6280	0.5729	0.5972	**0.6652**	**0.6444**
AMR	No-Attack	*0.9449*	*0.8342*	*0.5063*	*0.3876*	*0.4924*	*0.1070*
	FGSM ($\epsilon = 4$)	**1.3173**	0.7135	0.4565	0.3295	0.4332	0.4103
	FGSM ($\epsilon = 8$)	1.2814	0.7137	0.4429	0.3053	0.4318	0.4007
	PGD ($\epsilon = 4$)	1.1958	0.6473	0.4900	0.8064	0.4435	0.4173
	PGD ($\epsilon = 8$)	1.2377	0.6770	0.4445	**2.1264**	0.4323	0.3942
	C&W	1.3012	0.7159	0.4977	0.3610	0.4293	**0.4378**

training strategies have failed in protecting VBPR since the data poisoning is always effective in any defended settings.

Investigating AMR results, the attacks are quite effective in the defense-free settings as much as in VBPR, and confirm PGD ($\epsilon = 8$) as the most powerful method. Interestingly, the joint usage of (1) adversarial training procedures on the DNN and (2) the adversarial regularization on the recommender embeddings (APR) significantly reduced the effectiveness of the dataset poisoning. Indeed, 75% of attacks have not increased the Category Hit Ratio @ 20 of the low popular category of products.

Visual Evaluation

To investigate the efficacy of attacks in poisoning the VRS, we studied the attack Success Rate (SR), the Feature Loss (FL), and the Learned Perceptual Image Patch Similarity (LPIPS) [386]. Given the importance that visual features hold in VRSs, FL calculates the MSE between extracted features before and after the attack. That is, it provides a measure of visual features' shifting in the latent space, and how this has affected recommendation. The idea behind LPIPS is to produce a perceptual distance value between two similar images by leveraging (1) knowledge extracted from convolutional layers inside state-of-the-art CNNs and (2) collected human visual judgments about those pairs of similar images. We computed this metric fine-tuning a VGG [284] network since Zhang et al. [386] proposed this configuration as the best one at imitating a real human-evaluation in circumstances comparable to visual attacks.

Table 5.9 reports the LPIPS results, along with SR and FL values. It is worth recalling that a large (small) FL value stands for *semantically* different (similar) images from DNN's point of view. Similarly, a large (small) LPIPS value means the two compared images would likely be considered as *visually* different (similar) by humans.

Two general observations arise here. First, the FL is strictly correlated to the SR, i.e., *an attack is successful when the extracted features are noticeably shifted in the latent space*. Second, all attack combinations are able to keep LPIPS values within low ranges, in accordance with the *imperceptible* nature of adversarial perturbations on images [299]. Thus, we connect this obtained measure with the attack efficacy in both failing the classifier (i.e., the DNN) and the VRS. What follows is a detailed evaluation of scenarios involving (or not) defensive techniques for the DNN.

Defense-free Setting. In the defense-free scenario, PGD ($\epsilon = 4$) is the least perceptible attack (with the lowest LPIPS values) even considering a near-100% SR and a successful pushing of attacked products. On the other hand, FGSM ($\epsilon = 8$) fails to hide the produced perturbations, reaching the highest perceptible visual difference on `Amazon Women` (2.8505). Coherently, this setting also shows a low SR and a weak alteration of visual recommendations (see Table 5.8).

Defense Setting. Let us focus on the two defenses. Here, it becomes fundamental to consider the LPIPS value along with its corresponding SR and recommendation variations. As a matter of fact, in a defense context, where all attacks averagely tend to perform worse at failing the DNN classifier, a measured low average LPIPS value might trivially mean *very few* images were *successfully* attacked. For instance, the described situation occurs in the combination <`Amazon Men`, PGD ($\epsilon = 4$), Adversarial Training>. However, since these attacks have still been effective in *pushing* low ranked category products (as evident in Table 5.8), then adversaries could exploit their hardly-human perceptibility to craft even stronger perturbations (e.g., increasing ϵ). An intriguing situation is when LPIPS on the defended DNN is higher than the non-defended one. The worst case is <`Amazon Men`, FGSM ($\epsilon = 8$), Adversarial Training>, which shows a 34% increase of LPIPS compared to the Traditional training. We explain this result considering that and attack might need to produce larger perturbations to move the category of the few correctly attacked images (about 24% in the cited example) towards the targeted one. Not only is the attack inefficient, but it risks human identification.

Table 5.9 Average values of Success Rate (SR), Feature Loss (FL) and Learned Perceptual Image Patch Similarity (LPIPS) for each <dataset, attack, defense> combination. LPIPS is multiplied by 100. We mark in **bold** the best results for each considered metric.

		Image Feature Extractor								
Dataset	**Attack**	**Traditional**			**Adversarial Training**			**Free Adversarial Training**		
		SR	FL	LPIPS	SR	FL	LPIPS	SR	FL	LPIPS
`Amazon Women`	FGSM ($\epsilon = 4$)	17.70%	0.0096677	0.2388	0.00%	0.0000113	0.1353	0.00%	0.0000094	0.1041
	FGSM ($\epsilon = 8$)	28.32%	0.0220499	2.8505	2.65%	0.0000851	1.8298	0.00%	0.0000671	1.2119
	PGD ($\epsilon = 4$)	84.96%	0.0276645	**0.1860**	0.00%	0.0000119	0.1093	0.00%	0.0000102	0.0860
	PGD ($\epsilon = 8$)	**100.00%**	**0.1303309**	1.1136	3.54%	0.0000974	0.7683	0.00%	0.0000735	0.6369
	C & W	89.38%	0.0212380	0.2678	**6.19%**	**0.0001770**	**0.0731**	**6.19%**	**0.0003376**	**0.0816**
`Amazon Men`	FGSM ($\epsilon = 4$)	65.45%	0.0140948	0.1861	18.32%	0.0000330	0.1407	15.18%	0.0000278	0.1074
	FGSM ($\epsilon = 8$)	86.91%	0.0363190	1.7124	23.56%	0.0002658	2.2903	20.42%	0.0002320	1.2293
	PGD ($\epsilon = 4$)	96.86%	0.0368843	**0.1669**	18.32%	0.0000334	**0.1257**	15.18%	0.0000283	**0.0892**
	PGD ($\epsilon = 8$)	**100.00%**	**0.1349854**	0.6916	24.08%	0.0002801	0.7997	20.94%	0.0002371	0.6468
	C & W	89.01%	0.0205172	0.2279	**48.17%**	**0.0028022**	0.2688	**42.41%**	**0.0019080**	0.1490

5.5 Summary

This chapter was devoted to analyzing and proposing novel solutions to the technical challenges recognizable in multimedia recommendation (and outlined in the previous chapter) in the particular setting of visually-aware recommender systems. First, a unified framework for the extraction of multimodal features in recommendation (Ducho) was proposed. Second, to seek reproducibility in visual-based recommender systems, V-Elliot (an extension of the Elliot framework) was introduced. The two frameworks were later exploited to benchmark the performance of visually-aware recommender systems with a number of pre-trained visual extractors, highlighting how deeper visual feature extractors (i.e., ResNet50) may provide improved recommendation performance on accuracy and beyond-accuracy measures, both qualitatively and quantitatively. Later, other outlined technical challenges were addressed by considering two scenarios and tasks in visually-aware recommendation: (i) fashion recommendation and (ii) adversarial attacks/defenses against visually-aware recommender systems. In terms of (i), we proposed a novel approach able to disentangle the users' preferences at the granularity of content-style properties of fashion products, outperforming other recommendation baselines on a number of accuracy and beyond-accuracy recommendation measures; moreover, an ablation study further motivated the design choices for our proposed approach. Regarding (ii), an investigation on the effects of adversarially-attacked product images for visual-based recommendation, along with defensive countermeasures, shed light on how such attacks may be perceived by the human customers on a number of computer vision metrics; specifically, we demonstrated the alarming weakness of adversarial training in protecting the recommendation performance, while the visual evaluation suggested defense scenarios with few successfully attacked images and barely perceptible visual artifacts that still keep breaking recommendation performance are blind spots that adversaries could explore deeper for their malicious purposes.

This chapter, along with the previous one, conclude the thesis' section about recommendation approaches leveraging multimodal information. The next chapter will deal with the second core topic discussed in this thesis, namely, graph neural networks (GNNs)-based recommendation approaches.

Chapter 6

Evaluation of graph-based recommender systems

By leveraging the same experimental and evaluation paradigms adopted in the previous chapters of the thesis, the current chapter proposes a multi-sided analysis on the second main topic of this thesis work, namely: GNNs-based recommender systems. As such approaches embrace the family of recommendation models leveraging the collaborative filtering paradigm, and following the naming scheme proposed in state-of-the-art techniques from the literature, we introduce the term "graph collaborative filtering" to indicate the novel paradigm adopting graph neural networks for collaborative filtering. In the following, we first describe how we introduced six state-of-the-art graph collaborative filtering approaches into Elliot, and made it into an out-of-the-box application at the convenience of researchers and practitioners in the field. After that, we use this tool to rigorously reproduce the results of such approaches on a number of popular recommendation datasets. Our analysis helps uncovering unexpected insights, especially regarding the possible connection between graph topological properties and recommendation performance of graph-based recommender systems. Indeed, we decide to conduct an in-depth study on this aspect, which sheds light on possible re-interpretations of topological properties of the user-item graph under the recommendation perspective, thus questioning the current architecture and strategies in graph collaborative filtering. In the last part of the chapter, the focus is on another novel evaluation dimension of graph-based recommender systems. By recognizing a taxonomy categorization of approaches in the literature, according to which node representation and neighborhood exploration are the core strategy patterns which vary within the large plethora of solutions proposed so far, we evaluate the graph models' efficacy on a number of beyond-accuracy recommendation measures, accounting for novelty, diversity, and consumer/provider fairness, using a single and multi-objective evaluation setting.

6.1 Graph collaborative filtering within Elliot

Despite the outbreak of graph-based recommender systems in both academia and industry by surpassing traditional CF approaches, limited effort has been put into building *unified* and *comprehensive* frameworks to train and evaluate state-of-the-art models. Among the most noticeable mentions, we may recall RecBole [396, 397], which implements eight graph recommendation models for general recommendation (e.g., NGCF [325], LightGCN [126], DGCF [328], SGL [343], NCL [179], and SimGCL [371]). Recently, Zhu et al. [409] pave the way to a shared benchmarking pipeline for recommendation (i.e., BARS), and integrate thirteen models from the graph CF literature (besides some of the aforementioned models, they also reproduce, for instance, PinSage [369], DisenGCN [199], NGAT4Rec [285], GFCF [278], and UltraGCN [217]).

In this first part of the chapter, we show how to run extensive experimental settings for six popular graph collaborative filtering models (i.e., NGCF, LightGCN, DGCF, SGL, UltraGCN and GFCF) that we recently integrated into Elliot [11], our framework for recommender systems evaluation. Our contributions may be summarized as follows:

- Differently from the stable version of Elliot[1] which uses TensorFlow as the primary backend [12], we introduce PyTorch Geometric[2] (i.e., one of the most popular Python libraries for geometric deep learning) as the additional backend to design graph-based baselines adopting the explicit message-passing schema; at the time of this publication, only a few other frameworks have started to adopt it [397].
- Given the known reproducibility issues related to some non-deterministic operations in PyTorch Geometric [272], we implement message-passing with sparse adjacency matrices [219].
- In contrast to existing similar solutions (i.e., RecBole and BARS), we implement six state-of-the-art graph baselines by following a novel model categorization [17, 18] that distinguishes between methods using explicit message aggregation (i.e., NGCF, LightGCN, DGCF, and SGL) and going beyond the concept of graph convolution (i.e., UltraGCN and GFCF).

[1]https://github.com/sisinflab/elliot.
[2]https://pytorch-geometric.readthedocs.io/en/latest/.

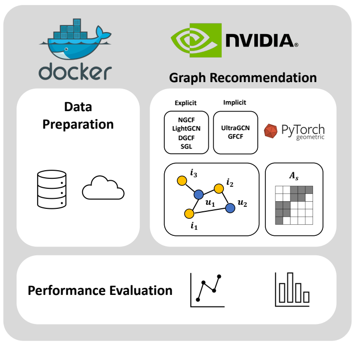


Fig. 6.1 Architecture of Elliot for graph collaborative filtering. We integrate PyTorch Geometric as backend, categorize graph models into two classes, and dockerize the application.

- As we leverage GPU boost by running PyTorch baselines with CUDA, we prepare a Docker image[3] creating a self-consistent and out-of-the-box experimental environment that requires minimal third-party libraries installed on the local machine. To the best of our knowledge, Elliot is the first recommendation framework to offer such functionalities.

Codes, datasets, and a video tutorial to install and launch the application are accessible at a public GitHub repository[4].

6.1.1 Proposed application

This section presents our application for graph collaborative filtering in Elliot. First, we focus on the integration of PyTorch Geometric by addressing its reproducibility issues. Then, we describe the complete procedure to dockerize our application. Finally, we outline the steps to easily install and train/evaluate a graph recommender.

PyTorch Geometric in Elliot. Figure 6.1 depicts the overall architecture for our framework, organized into: (i) *data preparation*, (ii) *recommendation*, and (iii) *performance evaluation*. Diving into each of these modules is out of the scope of this paper (we extensively explained them in previous works [11, 12]). Conversely, our main focus is on integrating PyTorch Geometric into the existing Elliot environment to build and run graph recommendation models. It is worth mentioning that, in contrast to other graph recommendation frameworks, we propose a novel model categorization [17, 18] where we consider (i) explicit message-passing (e.g., LightGCN) and (ii) the simplification of graph convolution (e.g., UltraGCN). In the following, we deepen into graph convolution for recommendation.

[3]https://hub.docker.com/r/sisinflabpoliba/demo-graph.
[4]https://github.com/sisinflab/Graph-Demo.

Given a user and item embeddings $\mathbf{e}_u$ and $\mathbf{e}_i$, the general formulation for the message-passing schema after l hops is:

$$\mathbf{e}_u^{(l)} = \omega\Big(\Big\{\underbrace{\mathbf{e}_{i'}^{(l-1)}, \forall i' \in \mathcal{N}(u)}_{\text{messages}}\Big\}\Big), \quad \mathbf{e}_i^{(l)} = \omega\Big(\Big\{\underbrace{\mathbf{e}_{u'}^{(l-1)}, \forall u' \in \mathcal{N}(i)}_{\text{messages}}\Big\}\Big), \tag{6.1}$$

where $\omega(\cdot)$ is the message aggregation, while $\mathcal{N}(u)$ and $\mathcal{N}(i)$ are the sets of 1-hop neighbor nodes for u and i.

Graph approaches leveraging message-passing (i.e., NGCF, LightGCN, DGCF, and SGL) inherit the **`MessagePassing`** base class [251]. This class provides, among the others, the functions **`propagate`** (which performs both the **`message`** and **`aggregate`** operations, defining the generic message and the message aggregation, respectively) and **`forward`** (which generates the outputs). The required input format to the **`forward`** function are the node embeddings at hop $l-1$ ($\mathbf{E}^{(l-1)}$) and an edge array of dimension $2 \times 2M$ (**edges**), with M as the number of user-item/item-user recorded interactions, storing indices of users and items with a **bidirectional** connection.

Such implementation is straightforward, especially because it permits explicitly defining the custom message formulation for the generic node as done in several works from the literature (refer again to Equation (6.34)). However, there are known reproducibility issues [272] related to some operations in PyTorch Geometric since it **could** behave non-deterministically (e.g., the **`scatter`** function, called by **`aggregate`**). To handle it, we follow one of the most common strategies [272]. That is, we reformulate the single node message-passing schema from Equation (6.34) into a matrix formulation adopting sparse adjacency matrices [219]. As an example, let us define the message-passing formula for LightGCN [126] as:

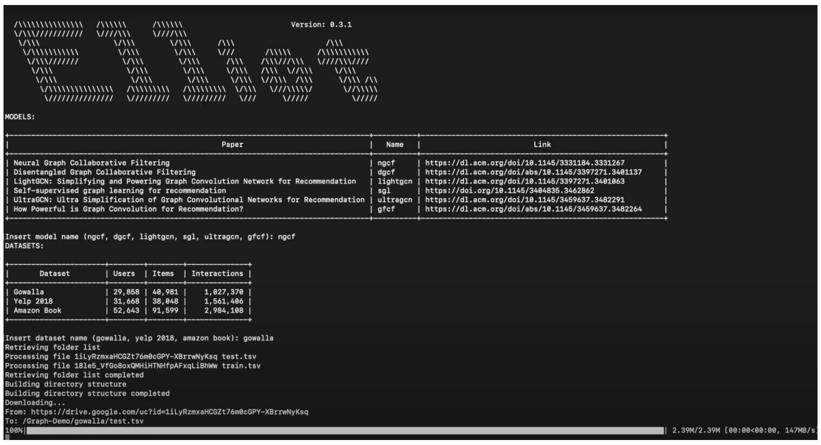
$$\mathbf{e}_u^{(l)} = \sum_{i' \in \mathcal{N}(u)} \mathbf{e}_{i'}^{(l-1)}, \quad \mathbf{e}_i^{(l)} = \sum_{u' \in \mathcal{N}(i)} \mathbf{e}_{u'}^{(l-1)}. \tag{6.2}$$

We may rewrite it into the following compact expression:

$$\mathbf{E}^{(l)} = \mathbf{A}_s \mathbf{E}^{(l-1)}, \tag{6.3}$$

where $\mathbf{A}_s$ is the sparse adjacency matrix for the bipartite and undirected user-item graph. By passing $\mathbf{A}_s$ instead of **edges** as input to **`forward`**, it will trigger the call of the **`message_and_aggregate`** function, which does not make use of the **`scatter`** operation. Unfortunately, this workaround is not always feasible (e.g., DGCF [328]).

Dockerization. As in similar Python frameworks for machine and deep learning (e.g., TensorFlow), PyTorch Geometric also supports models' training and inference with CUDA technologies for NVIDIA GPUs. Setting up a suitable development environment



```
/\\\\\\\\\\\\\\\   /\\\\\\     /\\\\\\                         Version: 0.3.1
\/\\\///////////   \////\\\    \////\\\
 \/\\\                \/\\\       \/\\\     /\\\           /\\\
  \/\\\\\\\\\\\        \/\\\       \/\\\    \///     /\\\\\     /\\\\\\\\\\\
   \/\\\///////         \/\\\       \/\\\     /\\\  /\\\///\\\  \////\\\////
    \/\\\                \/\\\       \/\\\    \/\\\ /\\\  \//\\\    \/\\\
     \/\\\                \/\\\       \/\\\    \/\\\ \//\\\  /\\\     \/\\\ /\\
      \/\\\\\\\\\\\\\\\  /\\\\\\\\\  /\\\\\\\\\ \/\\\  \///\\\\\/      \//\\\\\
       \///////////////  \/////////  \/////////  \///     \/////         \/////
MODELS:

+-----------------------------------------------------------------------------------+----------+---------------------------------------------------+
|                                       Paper                                       |   Name   |                        Link                       |
+-----------------------------------------------------------------------------------+----------+---------------------------------------------------+
| Neural Graph Collaborative Filtering                                              | ngcf     | https://dl.acm.org/doi/10.1145/3331184.3331267     |
| Disentangled Graph Collaborative Filtering                                        | dgcf     | https://dl.acm.org/doi/abs/10.1145/3397271.3401137 |
| LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation   | lightgcn | https://dl.acm.org/doi/10.1145/3397271.3401063     |
| Self-supervised graph learning for recommendation                                 | sgl      | https://doi.org/10.1145/3404835.3462862            |
| UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation | ultragcn | https://dl.acm.org/doi/10.1145/3459637.3482291     |
| How Powerful is Graph Convolution for Recommendation?                             | gfcf     | https://dl.acm.org/doi/abs/10.1145/3459637.3482264 |
+-----------------------------------------------------------------------------------+----------+---------------------------------------------------+

Insert model name (ngcf, dgcf, lightgcn, sgl, ultragcn, gfcf): ngcf
DATASETS:

+----------------------+--------+--------+--------------+
|       Dataset        | Users  | Items  | Interactions |
+----------------------+--------+--------+--------------+
| Gowalla              | 29,858 | 40,981 |    1,027,370 |
| Yelp 2018            | 31,668 | 38,048 |    1,561,406 |
| Amazon Book          | 52,643 | 91,599 |    2,984,108 |
+----------------------+--------+--------+--------------+

Insert dataset name (gowalla, yelp 2018, amazon book): gowalla
Retrieving folder list
Processing file 1iLyRzmxaHCGZt76m0cGPY-XBrrwNyKsq test.tsv
Processing file 18le5_VfGo8oxQMHiHTNHfpAFxqLiBhWw train.tsv
Retrieving folder list completed
Building directory structure
Building directory structure completed
Downloading...
From: https://drive.google.com/uc?id=1iLyRzmxaHCGZt76m0cGPY-XBrrwNyKsq
To: /Graph-Demo/gowalla/test.tsv
100%|██████████████████████████████████████████████████████████████████████| 2.39M/2.39M [00:00<00:00, 147MB/s
```

Fig. 6.2 Screenshot of the application start, where user can select the model (e.g., NGCF) and the dataset (e.g., Gowalla).

where versions compatibility is ensured for CUDA, cuDNN, and other third-party libraries could be cumbersome in some cases, especially in scenarios where users may need different installed versions of the same frameworks and libraries on one workstation. To tackle this challenge, we decide to leverage Docker[5] to create a self-consistent and out-of-the-box environment that provides the necessary libraries already installed in a proper configuration setting. Quite conveniently, we adopt the Docker container toolkit provided by NVIDIA[6] for the creation of containers equipped with customizable versions of CUDA and cuDNN.

First, we build a Docker image derived from this NVIDIA image[7], which comes with Ubuntu 20.04, CUDA 11.6.2, and cuDNN 8. Additionally, the custom image includes other useful Linux packages (e.g., Python 3.8 and pip), a cloned version of our GitHub repository for this demonstration, and all required Python packages to run the framework. You may refer to this link for the Dockerfile we use to build the image. Finally, to pull and run a Docker container from it, we also release the docker-compose YAML file (accessible at this link). It allows all GPUs on the machine to be used within the container, creates a bind mount between the `results/` folder in the container and a homonym folder on the host (thus storing files permanently), and runs the application.

[5]https://www.docker.com/.
[6]https://github.com/NVIDIA/nvidia-docker.
[7]https://dockr.ly/3aWAtWt. The URL has been shortened to save space.

Installation and running. Thanks to the core functionalities of NVIDIA-powered Docker containers, the installation requirements needed to run our application are minimal. You may refer to the official installation guide[8].
As for the pre-requisites, ensure you have the proper NVIDIA drivers installed on the host machine. Then, follow the indicated procedure to install Docker[9] and the NVIDIA Container Toolkit[10]. If everything works smoothly, you should be able to pull and run a Docker container with any CUDA and cuDNN versions (you may test the application by launching a container with the command `nvidia-smi` which shows a snapshot on the GPU usage). Finally, Docker Compose needs to be installed on the host machine. Once everything has been correctly set up, the custom Docker image can be pulled, and a container can be instantiated from it. To do so, you may use the docker-compose YAML file we provide in the GitHub repository. When the application starts (Figure 6.2), the user is asked to insert the model's name and the dataset, that is downloaded on-the-fly from the cloud. After that, the selected model is trained, validated, and tested on the chosen dataset for some hyper-parameter configurations (see later). You may refer to the official Elliot's documentation[11] and GitHub page for a comprehensive presentation of the formatting of the results. We also release a video tutorial for the reader[12].

6.2 Reproducing and benchmarking graph-based recommender systems

In recent years, great effort has been devoted in creating GNN-based models that address the critical issues of existing models, such as the over-smoothing phenomenon [54] and scalability issues [369]. These cutting-edge models are taking the world of recommender systems by storm and ushering in a new era of accuracy [185, 217, 248, 278, 348]. Over the past ten years, the application of neural techniques rooted in graph representation learning, such as graph convolutional networks [158] (GCNs), has introduced a fresh perspective on traditional collaborative filtering (CF) approaches. Rather than relying solely on user-item interactions for optimization [128, 160, 258], GCN-based

[8]https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html.
[9]https://docs.docker.com/engine/install/ubuntu/.
[10]https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html#setting-up-nvidia-container-toolkit.
[11]https://elliot.readthedocs.io/en/latest/.
[12]https://www.youtube.com/watch?v=_Bpgf4wnwIU.

methods enable the extraction of both short- and long-distance user preferences toward items [325]. By incorporating multi-hop relationships into the embeddings of users and items, these learned profiles yield more precise recommendations, as evidenced in the literature [126, 217]. Nevertheless, more researchers obtained different accuracy outcomes in independent experiments and began questioning the graph collaborative filtering (graph CF) prominence [409].

The original GCN layer employs message-passing techniques to refine the node representations of users and items through the iterative aggregation of their respective multi-hop neighbor nodes. While early attempts focused on simple aggregation methods [32, 369], recent solutions have advanced the field by exploring the interdependencies between nodes and their neighbors [325], designing simplified versions of the graph convolutional layer [61, 126] and learning multiple nodes' views [343, 371] augmented via self-supervised and contrastive learning to improve model accuracy. Moreover, current trends aim to simplify message-passing formulations [217, 248, 278], explore other spaces for graph-based recommendation tasks [278, 290, 388], and use hypergraphs to capture complex user-item dependencies [333, 349]. To filter out noisy neighbors and uncover hidden preference patterns, a complementary research field emerged that focuses on learning importance weights through attention mechanisms, such as those employed in the graph attention network [313] (GAT). While some models aim to recognize meaningful user-item interactions at a higher level [305, 324], others disentangle relations on a finer-grained scale [328, 388]. The recent advancements in GCN-based techniques have opened up new avenues for more accurate and effective recommendation systems.

Reproducibility is the cutting-edge research task in which researchers replicate experimental results using the same data and methods [31, 79, 80, 296]. In the case of graph CF, several factors contribute to the lack of reproducibility. Firstly, many graph CF studies copy previous results found in the literature for the same datasets, which makes it challenging to compare and reproduce results across different studies. Secondly, such studies do not provide the implementation of the adopted baselines, which makes it difficult to assess the effectiveness of different models. Furthermore, graph CF studies frequently do not provide complete information since they do not always share the experimental setups, such as hyper-parameter settings and training procedures. This lack of transparency makes it challenging to reproduce results and verify the validity of the findings. The lack of reproducibility in graph CF is a significant issue because it undermines the research's credibility and hinders the field's progress. To address this problem, researchers should strive to provide more detailed descriptions of their experimental setups and make their code and datasets publicly available. Additionally, the research community should work together to establish standard evaluation metrics and experimental protocols to promote reproducibility and facilitate comparison across different studies.

To this aim, this part of the chapter reports on a notable reproducibility effort to re-implement and replicate the results of six state-of-the-art (both well-established and recent) papers on graph collaborative filtering, namely, NGCF [325], DGCF [328], LightGCN [126], SGL [343], UltraGCN [217], and GFCF [278]. In particular, we provide an in-depth experimental analysis of the papers, conducting the experiments from scratch on the three datasets adopted in the original papers: Gowalla [176], Yelp 2018 [126], and Amazon Book [124]. Notably, the investigation extends the previous works by incorporating state-of-the-art classical collaborative filtering baselines such as UserKNN [259], ItemKNN [271], RP$^3\beta$ [246], and EASE^R [289] to correctly position the graph CF methods in the recommender systems state-of-the-art.

The study's findings reveal that RP$^3\beta$ ranks as the second-best method with the Yelp 2018 dataset, indicating that the original papers would have needed a more comprehensive evaluation. To this end, the evaluation benchmark incorporates two additional datasets, Allrecipes [102] and BookCrossing [412], which are common in the recommendation literature but uncommon in the graph CF-specific literature. However, surprisingly, the rankings significantly differ on the Allrecipes dataset, and the mathematical formulation of the graph CF methods is not sufficient to account for these outcomes. This observation leads to further investigation to comprehend the experimental results. Examining the dataset *topological* characteristics shows that the overall number of users and items and the average user and item degree vary from dataset to dataset. This observation may indicate the amount of information transmitted from node to node in the computational graph. According to the mathematical background, the analysis of the results is then threefold, focusing on the impact of (i) the *coldness/warmness* of a user, (ii) the *popularity* of the enjoyed items, and (iii) the *size* of the user *neighborhood* and the coldness/warmness of the neighbors. The users are partitioned in quartiles accordingly, and the experiments are re-evaluated to obtain more fine-grained results that motivate the outcomes for all the considered datasets.

Overall, the study aims to comprehensively answer several research questions, including:

RQ1. Is the state-of-the-art (i.e., the six most important papers) of graph collaborative filtering (graph CF) replicable?

RQ2. How does the state-of-art of graph CF position with respect to classic CF state-of-the-art?

RQ3. How does the state-of-art of graph CF perform on datasets from different domains and with different topological aspects, not commonly adopted for graph CF recommendation?

RQ4. What information (or lack of it) impacts the performance of the graph CF methods across the various datasets?

The following introduces the background and the experiments to answer the outlined research questions. First, in Section 6.2.1, we present the background technologies and the reproducibility details to conduct our study. Then, in Section 6.2.2, we report the reproducibility results, whose insights are complemented by adding novel classic CF baselines (i.e., Section 6.2.3). Furthermore, an investigation upon graph topology sheds light on the discrepancies of the graph CF approaches on two introduced datasets (i.e., Section 6.2.4). By reinterpreting the concept of users' node degree as *information flow* from the multi-hop neighborhoods to the user, we unveil the behavior of the graph and classic CF. Codes and datasets to reproduce our analysis are available here: https://github.com/sisinflab/Graph-RSs-Reproducibility.

6.2.1 Background and reproducibility analysis

The current section is aimed to provide the background about selected state-of-the-art methodologies in graph CF and their reproducibility details as presented in the original papers. First, the main aspects about graph-based models are introduced to conduct a chronological analysis of the strategies behind each algorithm. Then, we assess the experimental settings as reported in the original works by focusing on the chosen baselines, the datasets involved, and the training-testing protocol adopted in each case.

Graph collaborative filtering

In graph CF, users, items, and their interconnections are viewed as a bipartite and undirected graph. Let $\mathcal{U}$ and $\mathcal{I}$ be the sets of users and items in the recommendation system, respectively. Then, let $\mathbf{R} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ be the user-item interaction matrix where, in an implicit feedback scenario, $R_{ui} = 1$ if user $u \in \mathcal{U}$ interacted with item $i \in \mathcal{I}$, 0 otherwise. We build the adjacency matrix $\mathbf{A} \in \mathbb{R}^{(|\mathcal{U}|+|\mathcal{I}|) \times (|\mathcal{U}|+|\mathcal{I}|)}$ indicating the bi-directional connections linking users and items in $\mathbf{R}$:

$$\mathbf{A} = \begin{bmatrix} 0 & \mathbf{R} \\ \mathbf{R}^\top & 0 \end{bmatrix}. \tag{6.4}$$

We use the set of users and items, along with the adjacency matrix, to formally define the user-item bipartite and undirected graph $\mathcal{G} = \{\mathcal{U} \cup \mathcal{I}, \mathbf{A}\}$.

By associating users' and items' nodes to embeddings, the vast majority of approaches iteratively update their representations at different hop distances through the message-passing schema [42, 111].

For the second part of this chapter, we select and reproduce the results for six widely-recognized state-of-the-art approaches in graph CF, namely, NGCF [325], DGCF [328], LightGCN [126], SGL [343], UltraGCN [217], and GFCF [278] (refer to Section 6.2.2). This selection is motivated by two aspects: (i) such models are adopted as baselines in recent works from top-tier venues (see the second column in Table 6.8); (ii) their strategies cover a wide spectrum of techniques in graph CF. To provide a chronological overview of such techniques, in the following, we report their main aspects:

- **NGCF.** Neural graph collaborative filtering [325] (NGCF) is among the pioneer approaches in graph CF. Its message-passing schema works by aggregating the neighborhood information and the inter-dependencies among the *ego* and the *neighborhood* nodes (note that a normalized Laplacian adjacency matrix is used during the message-passing).

- **DGCF.** Disentangled graph collaborative filtering [328] (DGCF) assumes that user-item interactions can be disentangled into independent intents, where each stands for a specific aspect describing the user's preference towards the item. The model learns a set of weighted adjacency matrices refining the user-item importance related to a specific intent.

- **LightGCN.** Light graph convolutional network [126] (LightGCN) suggests that a more light-weight formulation of the graph convolutional layer proposed by Kipf et al. [158] can lead to superior accuracy performance in the recommendation scenario. Specifically, the architecture removes feature transformations and non-linearities.

- **SGL.** Self-supervised graph learning [343] (SGL) is among the first attempts to bring the lesson-learned from self-supervised [138] and contrastive [153] learning to graph CF. Built upon a LightGCN-based convolutional layer, the model learns different views of nodes by performing node/edge dropout and random walk operations on the graph topology. A self-supervised contrastive loss component is added to encourage the consistency among different views of the same node and the divergence among different nodes.

- **UltraGCN.** Ultra simplification of graph convolutional network [217] (UltraGCN) addresses some crucial issues in graph CF. Specifically, the authors propose a novel message-passing schema that mathematically approximates the infinite-layer propagation through a single (simplified) node update iteration. The adjacency matrix is normalized through a modified Laplacian formulation that accounts for the asymmetric weighting of connected nodes in user-user and item-item connections. Moreover, two loss components are introduced to tackle the over-smoothing effect and learn from the usually-unexplored type of node relationships such as item-item.
- **GFCF.** Graph filter-based collaborative filtering [278] (GFCF) questions the role of graph convolutional network into recommendation by leveraging graph signal processing theory. By showing that several existing approaches in CF may fall into one unified framework based upon graph convolution, the authors eventually propose a closed-form algorithm that proves to be a strong baseline against other trainable and computationally-expensive (graph-based) approaches in CF. Thus, the method represents the only exception to the message-passing models presented.

Analysis on reported baselines

Table 6.1 reports on the baselines each graph-based approach was tested against in the original paper. By categorizing them into *classic* and *graph* CF we first observe that, with the only exception of UltraGCN, all graph-based recommendation systems are generally compared only against 1-2 classical CF solutions (MF [128, 258]- and/or VAE [177, 200]-based approaches in most cases). However, the recent literature [14, 15, 79, 80, 409] has raised several concerns about usually-untested strong CF baselines, such as nearest-neighborhood approaches (e.g., UserKNN [259] and ItemKNN [271]), random-walk techniques (e.g., RP$^3\beta$ [246]), and other autoencoder-based solutions (e.g., EASE^R [289]). Differently from the classical CF baselines, we notice that most of the works compare their proposed approaches against a wide (and shared) range of graph CF solutions. This is easily explainable given the conceptual and logical similarities among the graph CF baselines and the proposed approaches. Moreover, besides a limited subset of graph CF baselines (i.e., HOP-Rec [355] and GRMF [253]), the vast majority of tested graph algorithms [32, 61, 199, 291, 369] are based upon the graph convolutional network architecture. Interestingly, we observe that only a subgroup of

Table 6.1 Analysis of baselines used in each of the selected graph-based models, categorized into *classic* and *graph* CF. A colored tick '✓' denotes when one of the baselines is also among the selected set of graph-based approaches for our study.

		Models					
Families	**Baselines**	NGCF [325]	DGCF [328]	LightGCN [126]	SGL [343]	UltraGCN [217]	GFCF [278]
		Used as graph CF baseline in (2021 — present)					
		[47, 56, 139, 290, 335, 358]	[96, 179, 216, 329, 331, 388]	[182, 254, 343, 349, 370, 371]	[103, 216, 335, 349, 363, 395]	[92, 113, 190, 237, 408, 409]	[17, 18, 185, 247, 347, 409]
Classic CF	MF-BPR [258]	✓	✓			✓	
	NeuMF [128]	✓					
	CMN [93]	✓					
	MacridVAE [200]		✓				
	Mult-VAE [177]			✓	✓		✓
	DNN+SSL [364]				✓		
	ENMF [52]					✓	
	CML [134]					✓	
	DeepWalk [249]					✓	
	LINE [301]					✓	
	Node2Vec [117]					✓	
	NBPO [376]					✓	
Graph CF	HOP-Rec [355]	✓					
	GC-MC [32]	✓	✓				
	PinSage [369]	✓					
	NGCF [325]		✓	✓	✓	✓	✓
	DisenGCN [199]		✓				
	GRMF [253]			✓			✓
	GRMF-Norm [126]			✓			✓
	NIA-GCN [291]					✓	
	LightGCN [126]				✓	✓	✓
	DGCF					✓	
	LR-GCCF [61]					✓	
	SCF [398]					✓	
	BGCF [292]					✓	
	LCFN [375]					✓	

our selected six graph CF approaches (up to a maximum of three approaches if we consider UltraGCN) is generally compared against the proposed approach. While we could justify this point with chronological motivations (e.g., DGCF could have not been tested on SGL, UltraGCN, and GFCF), we deem this to be an important lack in the existing literature.

Under the above considerations, and differently from the previous works, we compare the accuracy performance of the selected six graph CF approaches against strong CF techniques (UserKNN, ItemKNN, RP$^3\beta$ and EASE^R), while providing a complete evaluation setting which involves all the selected graph methods, where they are put against one another (refer to Section 6.2.3). To our knowledge, this work is one of the first attempts [409] to fill this gap.

Analysis on reported datasets

Table 6.2 displays the datasets adopted to train and test the reviewed graph-based recommender systems, as reported in the original papers. Notably, we recognize a total of seven recommendation datasets spanning different domains such as social networks (i.e., Gowalla), points-of-interest (i.e., Yelp 2018), e-commerce (i.e., the Amazon product categories and Alibaba-iFashion), and movies (i.e., Movielens 1M). It is worth pointing

Table 6.2 Analysis of the datasets adopted in each graph-based approach.

Models	Gowalla	Yelp 2018	Amazon Book	Alibaba-iFashion	Movielens 1M	Amazon Electronics	Amazon CDs
NGCF	✓	✓	✓				
DGCF	✓	✓	✓				
LightGCN	✓	✓	✓				
SGL		✓	✓	✓			
UltraGCN	✓	✓	✓		✓	✓	✓
GFCF	✓	✓	✓				

out that when we set the '✓' for the same dataset on different models, we are stating that the authors from the original works used the exact same dataset setting, that is, the original user-item interaction data and splitting/filtering strategies. A deeper analysis shows that there exists a subset of three datasets (i.e., Gowalla [176], Yelp 2018 [126], and Amazon Book [124]) which is utilized in the majority of graph CF works. For the sake of reproducibility, we replicate the original results calculated on such datasets for the six graph CF approaches (although the SGL paper does not provide results on Gowalla). Given the limited set of shared datasets among all the approaches, we include novel, *never-investigated* datasets to assess if their recommendation accuracy remains consistent on other domains and/or topologies (refer to Section 6.2.4).

Analysis on experimental comparison

As a final analyzed dimension, we discuss the protocol for the experimental comparison between the baselines and the proposed approach in each selected work. Being the pioneer model in the domain, the authors from NGCF train all proposed baselines from scratch. In the DGCF paper, the authors directly report the results of some baselines which are shared with NGCF and train the other baselines from scratch. In a similar manner, the authors by LightGCN, SGL, and UltraGCN copy the result values from the original papers, while the remaining models are trained from scratch. Finally, the authors in GFCF reproduce LightGCN as the baselines are exactly the same.

With reference to the copy-paste of the baseline results, authors often justify this practice by claiming that the adopted experimental settings (in terms of dataset splitting/filtering) are equal to the ones adopted by their (graph) CF baselines. Indeed, it is also worth mentioning that authors are in some cases shared across the works under investigation.

To remove all doubts, and differently from the mentioned works, we re-implement all algorithms by carefully following their original codes, and train/evaluate them through Elliot [11, 210]. Our goal is to provide a fair and repeatable experimental environment for the selected graph CF approaches, by using the hyper-parameter settings as indicated in each paper and/or shared online code to assess to what extent we can reproduce the original results. The reader may refer to Section 6.2.2 for a whole description of our settings.

6.2.2 Replication of prior results

This section focuses on how the replication of the experiments from the six state-of-the-art papers on graph CF stated before has been set up. It starts by defining the evaluation protocol applied to compare these methods in their respective works. After that, we present our replication results.

Settings

The experimental setup adopted in the first part of this study is designed primarily to replicate the results of the models included in this analysis [126, 138, 199, 217, 278, 325]. As mentioned earlier, we use the three most common datasets in this scenario to show the results of our replicability study. Specifically, we use Gowalla, Yelp 2018, and Amazon Book as provided in the public repositories of NGCF[13] and LightGCN[14]. All the proposed models (except SGL) use the same datasets with the same filtering/splitting. The authors state that they adopt a random split based on the 80/20 hold-out (i.e., for each user, 80% of the interactions is used to create the training set, while the remaining 20% constitutes the test set). Thus, each user-item interaction is treated as positive; all others are considered unfavorable. In addition, the authors leave 10% of the training as a validation set for tuning the hyper-parameters. However, this portion of the dataset is not indicated in the papers' extra material.

The adopted evaluation protocol is shared across all the analyzed papers. The approach is known as all-unrated-item [288]; for each user, we retain all candidate items with whom she does not interact with in the training set. To measure the quality of recommendations, we use the Recall and the nDCG on the top-20 recommendation lists for each user.

Each work performs its own tuning of the hyper-parameters (the Recall@20 is used as validation metric), by reporting on the search hyper-parameter spaces. Moreover, the best configurations on each dataset are usually provided in the respective papers and/or repositories.

[13]https://github.com/xiangwang1223/neural_graph_collaborative_filtering.
[14]https://github.com/kuandeng/LightGCN.

Thus, we set the hyper-parameters on each model-dataset as the best ones declared by the authors. The configuration files to run such experiments is fully available at our GitHub repository: https://split.to/Graph-Reproducibility. The careful reader would notice that the results reported in Table 6.3 for NGCF (see the '**Original**' column) differ from those shown in the in-proceedings version [325]. The reason is that the authors modified and recalculated the results obtained for the model and baselines due to errors in the pre-processing of the Yelp 2018 dataset and in the calculation of the nDCG. Thus, for the sake of fair reproducibility, and only in this case, we consider the results reported in the arXiv (most updated) version of the paper [326].

Results

Table 6.3 compares the results reported by the six papers focused on our study with those obtained in our implementation (using the tuned parameters specified in each work, as explained before). The new experiments closely approximate the original ones, with the most significant performance shift being in the 10^{-3} order. There are no noticeable distinctions in metrics, dataset, or algorithm used.

More specifically, in an algorithm basis, we observe the performance of GFCF is the best replicated one. This might be due to form algorithm, hence, no perturbations from random initializations are expected. The rest of the approaches evidence a similar (high) level of replication, although the shift for NGCF and DGCF rarely achieves the 10^{-4} order for the two metrics in all the datasets. In any case, considering the random initializations and stochastic learning processes [148], our replication of these approaches could be considered a success. No significant differences were found among the three datasets. SGL was not originally reported for Gowalla, so it was omitted from the table as we compared reported results with our implementations using the same hyper-parameters. In summary, these results confirm that, as discussed before, even though authors of these papers re-used the performance values from other papers just by copy-pasting them, this did not hurt the reproducibility of these approaches. As previously stated, our assumption for this behavior (which is not a safe practice in general [31]) is that the experiments of the original papers were all comparable because some authors are shared across contributions, which should guarantee that the settings and implementations of the algorithms are the same.

Table 6.3 Results of our replicability study on Gowalla, Yelp 2018, and Amazon Book for the selected state-of-the-art graph-based recommender systems. We calculate the performance shift between our conducted experiments and the original ones (as reported in their papers). Note that models have been sorted out according to the chronological order.

Datasets	**Models**	**Ours**		**Original**		**Performance Shift**	
		Recall	nDCG	Recall	nDCG	Recall	nDCG
Gowalla	NGCF	0.1556	0.1320	0.1569	0.1327	$-1.3 \cdot 10^{-03}$	$-7 \cdot 10^{-04}$
	DGCF	0.1736	0.1477	0.1794	0.1521	$-5.8 \cdot 10^{-03}$	$-4.4 \cdot 10^{-03}$
	LightGCN	0.1826	0.1545	0.1830	0.1554	$-4 \cdot 10^{-04}$	$-9 \cdot 10^{-04}$
	SGL*	—	—	—	—	—	—
	UltraGCN	0.1863	0.1580	0.1862	0.1580	$+1 \cdot 10^{-04}$	0
	GFCF	0.1849	0.1518	0.1849	0.1518	0	0
Yelp 2018	NGCF	0.0556	0.0452	0.0579	0.0477	$-2.3 \cdot 10^{-03}$	$-2.5 \cdot 10^{-03}$
	DGCF	0.0621	0.0505	0.0640	0.0522	$-1.9 \cdot 10^{-03}$	$-1.7 \cdot 10^{-03}$
	LightGCN	0.0629	0.0516	0.0649	0.0530	$-2 \cdot 10^{-03}$	$-1.4 \cdot 10^{-03}$
	SGL	0.0669	0.0552	0.0675	0.0555	$-6 \cdot 10^{-04}$	$-3 \cdot 10^{-04}$
	UltraGCN	0.0672	0.0553	0.0683	0.0561	$-1.1 \cdot 10^{-03}$	$-8 \cdot 10^{-04}$
	GFCF	0.0697	0.0571	0.0697	0.0571	0	0
Amazon Book	NGCF	0.0319	0.0246	0.0337	0.0261	$-1.8 \cdot 10^{-03}$	$-1.5 \cdot 10^{-03}$
	DGCF	0.0384	0.0295	0.0399	0.0308	$-1.5 \cdot 10^{-03}$	$-1.3 \cdot 10^{-03}$
	LightGCN	0.0419	0.0323	0.0411	0.0315	$+8 \cdot 10^{-04}$	$+8 \cdot 10^{-04}$
	SGL	0.0474	0.0372	0.0478	0.0379	$-4 \cdot 10^{-04}$	$-7 \cdot 10^{-04}$
	UltraGCN	0.0688	0.0561	0.0681	0.0556	$+7 \cdot 10^{-04}$	$+5 \cdot 10^{-04}$
	GFCF	0.0710	0.0584	0.0710	0.0584	0	0

**Results are not provided since SGL was not originally trained and tested on Gowalla [343].*

6.2.3 Benchmarking graph CF approaches using alternative baselines

In line with recent reproducibility works (such as [79]) that evidenced certain problems regarding the choice and optimization of the baselines used for comparison, in this section we assess how graph CF approaches perform relatively to classical CF baselines. As in the previous section, we specify first how the experiments are prepared, and the corresponding results are shown and discussed later.

Settings

We expand our investigation by examining four *classic* CF models to enhance the replicability analysis. Specifically, we select four models whose accuracy performance has rarely been compared with the *graph-based* CF approaches replicated in this study. The decision to include UserKNN, ItemKNN, RP$^3\beta$, and EASE^R is purposeful. We refer to [80] and (more recently) [14], which demonstrated the competitiveness of these baseline models compared to more recent approaches when a shared benchmark for comparison is employed among all involved methodologies. Furthermore, we also consider two unpersonalized approaches (i.e., MostPop and Random). The two models act as benchmarks to assess the effectiveness of customized methods compared to a user-agnostic solution.

For a fair comparison, the configuration delineated herein elucidates how the four *classic* CF models are tuned following the exact same training/test splitting reported in Section 6.2.2 and the same experimental protocol. The only difference is that (for obvious reasons) we need to explore the hyper-parameters of each classic CF model introduced in the comparison. Similarly to what the authors do in the original graph CF works, we retain the 10% of the training to generate a validation set, but decide to explore 20 distinct configurations for each model through the state-of-the-art Tree-structured Parzen Estimator (TPE) hyper-parameter search [33]. For every model, the final results correspond to the accuracy measure on the test set by setting the hyper-parameter configuration providing the best Recall@20 results on the validation set. The complete configuration files to run the classic CF baselines are provided in our repository: https://github.com/sisinflab/Graph-RSs-Reproducibility.

Results

Table 6.4 shows the results of the graph CF models (as previously replicated in Table 6.3) with the additional baselines. First, it is worth noting that, even though none of these baselines gets the best results in any of the three datasets considered, they achieve the second-best performance in Yelp 2018 (refer to RP$^3\beta$ with nDCG).

Second, none of the models in the reference family achieve competitive performance. While this is expected for the Random algorithm, it is an indication that either none of these datasets evidence a strong popularity bias or (considering the way they were processed) such bias was removed.

Third, some of the classic CF approaches (such as RP$^3\beta$ and UserkNN in Gowalla, and RP$^3\beta$ and EASE^R in Yelp 2018) demonstrate better performance than some of the state-of-the-art graph CF methods, in particular, they perform better than NGCF, DGCF, and LightGCN. This result is in line with recent experimental comparisons [15, 19, 79] where these baselines outperform other methods based on matrix factorization or neural networks. Moreover, to some extent, the fact that some graph CF methods are outperformed should not be surprising, since, as shown in Table 6.1, none of these baselines were included in the original papers.

6.2.4 Extending the experimental comparison to new datasets

This section aims to provide a full picture from an experimental point of view on two new datasets: Allrecipes and BookCrossing. First, we introduce the experimental

Table 6.4 Graph-based CF solutions tested against unpersonalized (i.e., reference) and classical CF approaches on Gowalla, Yelp 2018, and Amazon Book. While results for the graph-based approaches have been directly reported from our reproducibility study (see above), classical CF recommender systems have been fine-tuned on the two datasets to find their best configurations. **Boldface** and underline refer to best and second-to-best values, respectively.

Families	**Models**	**Gowalla**		**Yelp 2018**		**Amazon Book**	
		Recall	nDCG	Recall	nDCG	Recall	nDCG
Reference	MostPop	0.0416	0.0316	0.0125	0.0101	0.0051	0.0044
	Random	0.0005	0.0003	0.0005	0.0004	0.0002	0.0002
Classic CF	UserKNN	0.1685	0.1370	0.0630	0.0528	0.0582	0.0477
	ItemKNN	0.1409	0.1165	0.0610	0.0507	0.0634	0.0524
	$RP^3\beta$	0.1829	0.1520	0.0671	$\underline{0.0559}$	0.0683	0.0565
	$EASE^R$ *	0.1661	0.1384	0.0655	0.0552	**0.0710**	$\underline{0.0567}$
Graph CF	NGCF	0.1556	0.1320	0.0556	0.0452	0.0319	0.0246
	DGCF	0.1736	0.1477	0.0621	0.0505	0.0384	0.0295
	LightGCN	0.1826	$\underline{0.1545}$	0.0629	0.0516	0.0419	0.0323
	SGL	—	—	0.0669	0.0552	0.0474	0.0372
	UltraGCN	**0.1863**	**0.1580**	$\underline{0.0672}$	0.0553	$\underline{0.0688}$	0.0561
	GFCF	$\underline{0.1849}$	0.1518	**0.0697**	**0.0571**	**0.0710**	**0.0584**

**Results for $EASE^R$ on Amazon Book are taken from BARS Benchmark [409].*

settings followed to obtain the results presented in the following section. Then, we discuss these results in more detail, aiming to explain the insights derived from them.

Settings

Motivated by the previous results, we further enrich our analysis by investigating the behavior of all tested models on two datasets that have never been considered in any previous study involving graph-based approaches for recommendation, namely, Allrecipes [102] and BookCrossing [412].

Table 6.5 shows some statistics of these datasets, where we purposely decide to report both the benchmarking datasets for graph CF (i.e., Gowalla, Yelp 2018, and Amazon Book) and the newly introduced ones. On the one hand, Allrecipes exhibits quite discordant characteristics compared to the other datasets. Although it has a comparable density, users are more numerous than items, with a much lower average user and item node degrees compared to the other standard graph CF datasets. On the other hand, BookCrossing displays the lowest ratio between the number of users to items across all datasets, and a much higher density than all the others. In summary, the newly introduced datasets serve as a foundation to assess the performance in different (and never-explored) *topological* settings for graph CF baselines.

To adhere to the experimental setup presented so far, we adopt the all-unrated-item evaluation protocol, and split the two datasets with a random hold-out solution,

Table 6.5 Statistics calculated on the training sets of Gowalla, Yelp 2018, Amazon Book, Allrecipes, and BookCrossing. We indicate the number of user-item interactions through' Edges' while 'Avg. Deg. (U)' and 'Avg. Deg. (I)' refer to users' and items' average node degree (i.e., average interaction number).

Statistics	**Gowalla**	**Yelp 2018**	**Amazon Book**	**Allrecipes**	**BookCrossing**
Users	29,858	31,668	52,643	10,084	6,754
Items	40,981	38,048	91,599	8,407	13,670
Edges	810,128	1,237,259	2,380,730	80,540	234,762
Density	0.0007	0.0010	0.0005	0.0010	0.0025
Avg. Deg. (U)	27.1327	39.0697	45.2241	7.9869	34.7590
Avg. Deg. (I)	19.7684	32.5184	25.9908	9.5801	17.1735

ensuring an 80:20 proportion. Differently from the replicability study, we now perform a TPE-based hyper-parameter tuning for *all* models, as the best hyper-parameters for each graph-based approach is not known in advance; for this, we (again) use the 10% portion of the training set as validation set. We run 20 different settings within the search space provided in the original papers. The models' best configurations are selected through the Recall@20 on the validation. As stated for previous settings, we report all configuration files to run the experiments with the novel datasets in the GitHub here: https://github.com/sisinflab/Graph-RSs-Reproducibility.

Results

Table 6.6 provides a full comparison between unpersonalized methods, classical CF approaches, and the graph CF methods under analysis. In line with our previous section experiments, classic CF methods (in particular, RP$^3\beta$ and EASE^R) are very competitive compared to graph CF approaches, even in novel datasets like the ones included in this analysis. More specifically, the results in BookCrossing are dominated by these baselines, whereas in Allrecipes, the MostPop technique is the one that stands out, evidencing a strong popularity bias.

These results highlight that, among the graph CF techniques, those that maintain their performance in novel domains are UltraGCN (best one in Allrecipes and third among its type) and LightGCN (second best in both domains). While the nature of these two datasets is clearly different (as shown in Table 7.10, Allrecipes is smaller and it contains more users than items, instead of the other way around as in BookCrossing), the relative performance of the best graph CF methods is competitive. However, for some of them, the performance drop is significant, reaching an accuracy lower than that of any other classic CF baseline.

Table 6.6 Graph-based CF solutions tested against unpersonalized (i.e., reference) and classical CF approaches on Allrecipes and BookCrossing. Boldface and underline refer to best and second-to-best values, respectively.

Families	**Models**	**Allrecipes**		**BookCrossing**	
		Recall	nDCG	Recall	nDCG
Reference	MostPop	<u>0.0472</u>	<u>0.0242</u>	0.0352	0.0319
	Random	0.0024	0.0010	0.0013	0.0011
Classic CF	UserKNN	0.0339	0.0188	0.0871	0.0769
	ItemKNN	0.0326	0.0180	0.0779	0.0739
	$RP^3\beta$	0.0170	0.0089	**0.0941**	<u>0.0821</u>
	$EASE^R$	0.0351	0.0192	<u>0.0925</u>	**0.0847**
Graph CF	NGCF	0.0291	0.0144	0.0670	0.0546
	DGCF	0.0448	0.0234	0.0643	0.0543
	LightGCN	0.0459	0.0236	0.0803	0.0660
	SGL	0.0365	0.0192	0.0716	0.0600
	UltraGCN	**0.0475**	**0.0248**	0.0800	0.0651
	GFCF	0.0101	0.0051	0.0819	0.0712

To bring light into some of these behaviors, the next section discusses in more detail how the ranking of the graph CF methods changes depending on the dataset, and hypothesize which dataset characteristics may be tied to these effects.

Discussion

To further validate and explain the reasons behind the results reported in Table 6.6, in the following we perform a twofold analysis. First, we rank all the selected graph-based recommendation models on all the tested datasets to assess their relative improvement across all settings and provide another perspective on the results from Table 6.6. Then, we propose a more nuanced study on the measured accuracy performance by investigating its (possible) dependence on the specific dataset characteristics, namely, the node degree as viewed at multiple hops.

Graph-based models' ranking. In Table 6.7, we rank the six graph CF recommender systems under analysis according to the calculated Recall@20 and nDCG@20, for both the original datasets (i.e., Gowalla, Yelp 2018, and Amazon Book) and the novel datasets we introduced (i.e., Allrecipes and BookCrossing). Moreover, we also indicate the relative improvement of each model with respect to the worst-performing algorithm on that dataset.

The trend on the three original datasets is quite steady, with UltraGCN and GFCF being the two best-performing approaches in almost all cases, and the remaining graph techniques ranked as in descending chronological order (confirming the findings from the recent literature). In terms of relative improvements, we observe large performance differences mainly on the Amazon Book setting.

Table 6.7 Graph-based recommender systems, ranked according to their Recall@20 and nDCG@20 on all the tested datasets. For each model, we also report its relative improvement with respect to the worst-performing approach on the same dataset (in green).

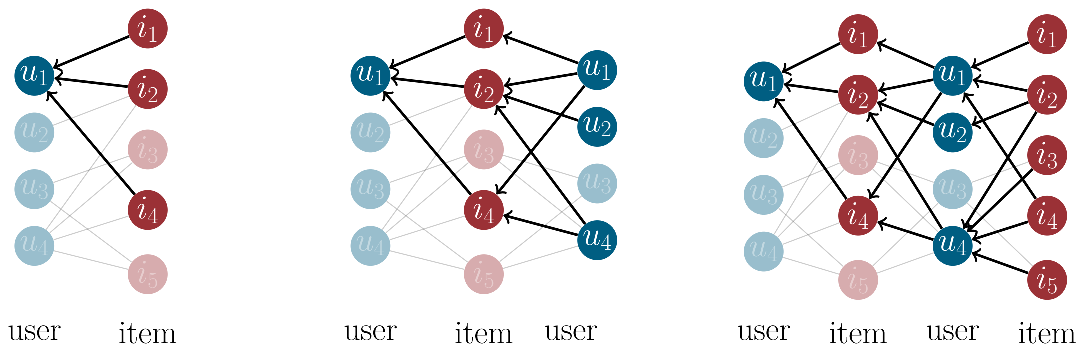
Metric		**Gowalla**	**Yelp 2018**	**Amazon Book**	**Allrecipes**	**BookCrossing**
Recall	**1.**	UltraGCN *(+19.73%)*	GFCF *(+25.36%)*	GFCF *(+122.57%)*	UltraGCN *(+370.30%)*	GFCF *(+27.37%)*
	2.	GFCF *(+18.83%)*	UltraGCN *(+20.86%)*	UltraGCN *(+115.67%)*	LightGCN *(+354.46%)*	LightGCN *(+24.88%)*
	3.	LightGCN *(+17.35%)*	SGL *(+20.32%)*	SGL *(+48.59%)*	DGCF *(+343.56%)*	UltraGCN *(+24.42%)*
	4.	DGCF *(+11.57%)*	LightGCN *(+13.13%)*	LightGCN *(+31.35%)*	SGL *(+261.39%)*	SGL *(+11.35%)*
	5.	NGCF (—)	DGCF *(+11.69%)*	DGCF *(+20.38%)*	NGCF *(+188.12%)*	NGCF *(+4.20%)*
	6.	*SGL** (—)	NGCF (—)	NGCF (—)	GFCF (—)	DGCF (—)
nDCG	**1.**	UltraGCN *(+19.70%)*	GFCF *(+26.33%)*	GFCF *(+137.40%)*	UltraGCN *(+386.27%)*	GFCF *(+31.12%)*
	2.	LightGCN *(+17.05%)*	UltraGCN *(+22.35%)*	UltraGCN *(+128.05%)*	LightGCN *(+362.75%)*	LightGCN *(+21.55%)*
	3.	GFCF *(+15.00%)*	SGL *(+22.12%)*	SGL *(+51.22%)*	DGCF *(+358.82%)*	UltraGCN *(+19.89%)*
	4.	DGCF *(+11.89%)*	LightGCN *(+14.16%)*	LightGCN *(+31.30%)*	SGL *(+276.47%)*	SGL *(+10.50%)*
	5.	NGCF (—)	DGCF *(+11.73%)*	DGCF *(+19.92%)*	NGCF *(+182.35%)*	NGCF *(+0.55%)*
	6.	*SGL** (—)	NGCF (—)	NGCF (—)	GFCF (—)	DGCF (—)

**SGL is not classifiable on the Gowalla dataset as results were not calculated in the original paper [343].*

By focusing on the two additional datasets (i.e., Allrecipes and BookCrossing), the rankings corroborate some of the previous outcomes, but also introduce novel and unexpected considerations. While UltraGCN seems to preserve its role of leading approach in the two scenarios (in BookCrossing it is ranked as third but with minimum margin to the second one), we notice how GFCF's performance is very fluctuating, as it even stands in the last position on Allrecipes with large performance difference to the other models (the same goes for DGCF). Noticeably, LightGCN gets up to the top of the ranking in both settings, indicating that a careful hyper-parameter tuning could be beneficial to outperform most of the other approaches, even the ones that should surpass it according to the literature (such as SGL). As final remarks, NGCF poor performance is again confirmed in such different dataset settings.

Analysis on the node degree. As already observed in Table 6.5, the average node degree of users and items represents one of the main aspects discerning each dataset from the other ones. For this reason, we decide to reason about its possible influence on the models' performance. In this respect, instead of limiting our analysis to the sole definition of node degree (i.e., number of recorded interactions for each user and item), and given the ability of graph-based approaches to distill the collaborative signal by stacking multiple layers [325], we propose a novel investigation which reinterprets the node degree as *information flow* from neighbor nodes to the user nodes after multiple hops. Note that we only consider users as the ending nodes of such a flow because we are interested in assessing how the accuracy recommendation measures (which are generally calculated user-wise) may be influenced by this aspect.

Before diving into the results and discussion, we provide some useful intuitions and formulations which may help understand our analysis. With reference to Figure 6.3, we introduce the definition of information flow at one, two, and three hops. We decide to limit our focus on the first three explored hops because (i) graph-based recommender



(a) 1-hop (b) 2-hop (c) 3-hop

Fig. 6.3 A toy user-item graph where the ego user node (highlighted) receives the *information flow* from the (a) 1-, (b) 2-, and (c) 3-hop neighbor nodes (highlighted). Arrows' direction is a visual representation of the information flow.

systems built upon the message-passing schema usually tend not to iterate over the third aggregation layer, and (ii) the investigation of more than three hops would not be meaningful from a recommendation perspective. As a matter of fact, we interpret each of the three hops as follows:

- at **one** hop (Figure 6.3a), users receive the information coming from the items they interacted with; in other words, this is an indication of the *activeness* of users on the platform;
- at **two** hops (Figure 6.3b), users receive the information of the other users co-interacting with the same items; in other words, this is an indication of the influence of *items' popularity* on users;
- at **three** hops (Figure 6.3c), users receive the information coming from the items interacted by the other users involved in co-interactions; i.e., this is an indication of the influence of *co-interacting* users' *activeness* on users.

Let us formalize such definitions. The information received by users at one, two, and three hops is calculated as:

$$\boldsymbol{\Upsilon}_{\mathcal{U}}^{(1)} = \mathbf{R}\mathbf{1}_{\mathcal{I}}, \qquad \boldsymbol{\Upsilon}_{\mathcal{U}}^{(2)} = (\mathbf{R} \odot (\mathbf{1}_{\mathcal{U}}\mathbf{R}))\mathbf{1}_{\mathcal{I}}, \qquad \boldsymbol{\Upsilon}_{\mathcal{U}}^{(3)} = (\mathbf{R}\mathbf{R}^\top \odot \mathbf{R}\mathbf{1}_{\mathcal{I}})\mathbf{1}_{\mathcal{I}}, \tag{6.5}$$

where $\boldsymbol{\Upsilon}_{\mathcal{U}}^{(h)} \in \mathbb{R}^{|\mathcal{U}|\times 1}$ is the vector of the information that all users receive from the nodes in their h-hop, $\mathbf{1}_{\mathcal{U}} \in \mathbb{R}^{1\times|\mathcal{U}|}$ and $\mathbf{1}_{\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|\times 1}$ are row and column vectors with 1 repeated $|\mathcal{U}|$ and $|\mathcal{I}|$ times, respectively, while $\odot$ is the Hadamard product performed in broadcast.

In light of the above, the study assesses the accuracy performance of graph-based recommender systems on user groups considering the information received from the one, two, and three hops neighborhood. Following other analyses in the literature, we decide to split users into quartiles according to the information values (i.e., $\boldsymbol{\Upsilon}_{\mathcal{U}}^{(h)}$). Thus, we consider four groups: (i) users whose values are below the 25% of the distribution, (ii) users whose values are above the 25% and below the 50% of the distribution, (iii) users whose values are above the 50% and below the 75% of the distribution, and (iv) users whose values are above the 75% of the distribution.

Figure 6.4 displays the percentage variation in accuracy performance (measured by nDCG) across quartiles relative to the average value reported in Table 6.6. The figure illustrates how the quality of recommendation performance fluctuates amongst different clusters of users. For example, a method indicating a 50% improvement in the fourth quartile would suggest that users in this cluster, typically more active (1-hop) or also interested in popular items (2-hop), receive more accurate recommendations with respect to the average user.

This observation implies that a non-discriminatory recommendation system should produce no variation across quartiles, with values overlapping the 0% dashed line. The second necessary preliminary to understand the outcome of the experiments is the interpretation of the quartiles for the different hops. In the 1-hop, the fourth quartile pertains to warm users interacting most with the platform, while the first quartile represents cold users interacting less frequently. In the 2-hop, high values in the fourth quartile indicate active users who enjoy popular items, resulting in dense subgraphs. The first quartile, in contrast, consists of less active users interacting with niche items in less dense subgraphs. The 3-hop, which includes user neighbors, generates the highest values when active users interact with popular items enjoyed by warm users (i.e., their neighbors). However, it is essential to note that the plots offer no insight into overall accuracy (which is in Table 6.6).

When considering the recommendation performance according to the corresponding cluster (depicted in Figure 6.4), it is crucial to note that none of them demonstrate ideal recommendation behavior. Instead, these systems tend to favor warm users or densely interconnected subgraphs located in the fourth quartile. Despite this trend, the 1-hop plots for graph Collaborative Filtering (CF) and classic CF methods in Allrecipes and BookCrossing graphs demonstrate minimal disparities between different recommendation approaches. Even though they all favor the fourth quartile over the first one, the coldness/warmness of a user marginally impacts how much the method is biased toward these types of users. The lone exception to this trend is GFCF, which exhibits even greater penalization towards the first three quartiles (varying on the three hops from, approximately, -45% to +115%, and thus exceeding the plots' upper

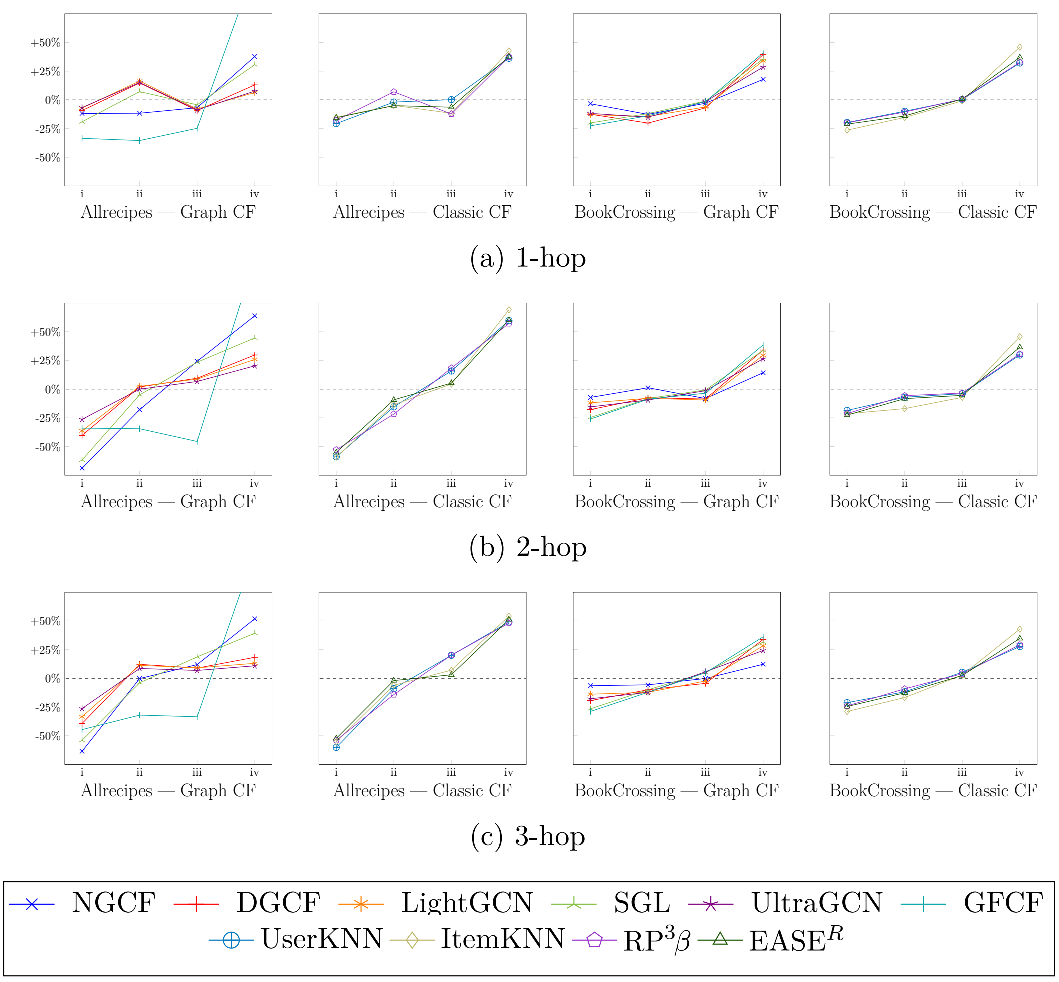


Fig. 6.4 Percentage variation between the nDCG on user quartiles and the average nDCG value across all users (indicated as the dashed line), for each model-dataset setting. Rows refer to user quartiles when considering (a) 1-, (b) 2-, and (c) 3-hop.

bound). As such, this system only provides satisfactory recommendation performance for users in the fourth quartile.

Regarding the 2-hop, there are several interesting insights to be gained. Firstly, the recommendation methods exhibit a higher overall slope, favoring the users who enjoyed popular items over the cold users who enjoyed niche items. While this may seem like an obvious observation, the plot confirms that user coldness/warmness alone is not a sufficient indicator of high-quality recommendations. Instead, the 2-hop reveals that combining user coldness/warmness and item popularity is useful for identifying such users. A second noteworthy aspect is that the Allrecipes dataset highlights three distinct behaviors among the graph CF methods. UltraGCN, DGCF, and LightGCN exhibit similar performance and display less discriminatory behavior across quartiles.

It is interesting to note that these models also perform best overall (see Table 6.6). On the other hand, SGL and NGCF show a higher slope that is comparable to classic CF.

Also, their corresponding performance is similar in Table 6.6. A third observation concerns GFCF, which performs poorly across all quartiles except for the fourth. Its behavior is even more accentuated than in the 1-hop analysis. Additionally, NGCF, SGL, and GFCF are graph CF algorithms performing differently according to user warmness and item popularity. Meanwhile, all algorithms in BookCrossing, and the classic CF in Allrecipes, exhibit the distribution over the quartiles across methods.

Finally, in the 3-hop, for the BookCrossing dataset, the information pertaining to neighbors does not contribute significantly to the results, as indicated by the similarity between the 2- and 3-hop plots. Meanwhile, in Allrecipes, the best models (UltraGCN, DGCF, and LightGCN) exhibit more consistency in performance across all quartiles, as demonstrated by a more even distribution of results (less variations across the quartiles). However, this pattern is not evident in NGCF, SGL, and GFCF, which exhibit a more disparate range of results across the quartiles.

6.3 A topology-aware analysis of graph collaborative filtering

From an ***algorithmic*** perspective, the technical contribution of graph-based recommender systems has been theoretically [278] and empirically [331] investigated to justify their high-quality recommendations. On the one hand, established approaches such as LightGCN [126] and DGCF [328] re-adapt the GCN layer to suit the collaborative filtering schema. Specifically, the former suggests that feature transformations and non-linearities should be removed since they could negatively impact the recommendation performance; the latter recognizes the importance of updating the user-item graph structure according to the learned users' intents towards items. On the other hand, recent graph-based RSs including UltraGCN [217] and SVD-GCN [248] acknowledge that existing graph-based models may still be affected by the over-smoothing phenomenon and scalability issues. To overcome such problems, they propose to go beyond the traditional concept of message aggregation at multiple layers by adopting ad-hoc mathematical proxies of the message-passing. UltraGCN approximates infinite propagation layers, while SVD-GCN explores the analogies between graph convolution and SVD. Additionally, both approaches discuss the importance of learning from user-user and item-item relationships.

From another perspective, the *machine learning* literature acknowledges that graph ***topology*** plays a crucial role in GNNs. The authors in [334] recognize how topology may increase the model's capacity by distinguishing between two approaches that influence the (topological) shape of GNNs. The first approach involves stacking aggregation operations, while the second one utilizes multiple aggregations of operations to prevent over-smoothing and enhance model capacity. Indeed, the learning ability of GNNs is a delicate balance between topology and node attributes. While the graph convolutional layer is gaining momentum due to Laplacian smoothing or low-pass filtering [342], stacking too many such layers can lead to a loss of expressive power, causing node representations to become dependent solely on node degree and connectivity. Two strategies are under debate to alleviate this issue [356]: modifying the learned representations [159, 394] (e.g., APPNP, GCNII, PairNorm) and modifying network topology [6, 159, 197, 262, 353] (e.g., DropEdge, GRAND, PPNP, MixHop, GDN, JKNet). In summary, the inconsistency between network topology and node content exists at the individual node and network levels. A recent study [280] suggests revisiting network topology to enhance GCN learning, but this involves revisiting the network topology thus modifying the underlying method.

In recommendation, some works have demonstrated how ***classical*** dataset statistics may impact the performance of recommendation models [8, 85, 266], for example, data sparsity within the user-item interaction matrix, the concentration of interactions from both user and item viewpoints or the ratio of users to items comprising the catalog. However, we believe that the topological nature of the user-item data (which allows us to interpret users/items and their interactions as a bipartite and undirected graph) along with the novel graph CF wave, could require a more careful analysis of additional (and less shallow) ***topological*** measures. By describing the topology of the user-item graph under the lens of its (multi)-hop connections, we regard it as imperative to unravel the interdependencies between topology-aware attributes and recommendation performance to better understand graph CF.

Motivated by the reasons above, we propose a topology-aware analysis of graph collaborative filtering to find the (possible) dependencies among (topological) data characteristics and recommendation performance of graph-based recommender systems. To this end, we first consider three popular datasets (i.e., Yelp2018, Gowalla, and Amazon-Book) and generate 1,800 synthetic sub-datasets with two common graph sampling strategies (i.e., node- and edge-dropout). Second, we select eleven *classical* and *topological* characteristics which are loosely correlated with one another and measure them for each of the sampled sub-datasets[15]. Then, we choose four graph-based RSs, namely, LightGCN [126], DGCF [328], UltraGCN [217], and SVD-GCN [248], since: (i) they are recent approaches, (ii) they are used as baselines in several works from the last years, (iii) they adopt different strategies that well depict most of the models from the literature. Last, an explanatory framework is trained to identify linear relations among characteristics and accuracy metrics. Our contributions are:

1. To the best of our knowledge, this is the first analysis of the influence of *classical* and *topological* dataset characteristics on the performance of state-of-the-art graph-based RSs, with a re-interpretation of the topology-aware characteristics under the lens of recommender systems.

2. We carefully choose four graph-based RSs that are across-the-board and recently proposed in the literature and also span an extensive selection of graph strategies. In an attempt to make the information conveyed by node degree explicitly emerge from their formulations, we aim to understand how each of them addresses such a topological aspect in all its facets.

3. We build an explanatory framework to calculate the linear dependencies among characteristics (the independent variables) and recommendation metrics (the dependent variables).

4. We validate the proposed framework on its statistical significance and uncover insights on graph CF under the novel perspective of graph topology. We further test the efficacy of the approach with varying settings of graph samplings, shedding light on the influence of node- and edge-dropout for our explanatory model.

Code and datasets to reproduce all results are available at: https://github.com/sisinflab/Graph-Characteristics.

6.3.1 Topological characteristics in recommendation data

By viewing the recommendation data as a bipartite and undirected user-item graph, we describe its *topological* characteristics [168, 226], which we re-interpret from the viewpoint of RSs.

[15]We indicate with *classical* those recommendation-based measures exploited in other similar works [8, 85], and with *topological* those properties of the user-item graph conceptually related to node degree [168, 226].

Preliminaries

In a recommendation system, we denote with $\mathcal{U}$ and $\mathcal{I}$ the sets of users and items, respectively, where $|\mathcal{U}| = U$ and $|\mathcal{I}| = I$. Then, we indicate with $\mathbf{R} \in \mathbb{R}^{U \times I}$ the interaction matrix collecting user-item interactions in the form of implicit feedback (i.e., $\mathbf{R}_{u,i} = 1$ if user $u \in \mathcal{U}$ interacted with item $i \in \mathcal{I}$, 0 otherwise). Moreover, let $\mathcal{N}_u = \{i \mid \mathbf{R}_{u,i} = 1\}$ and $\mathcal{N}_i = \{u \mid \mathbf{R}_{u,i} = 1\}$ be the sets of items and users having an interaction with u and i, respectively. We use $\mathbf{R}$ to define the adjacency matrix $\mathbf{A} \in \mathbb{R}^{(U+I)\times(U+I)}$ representing the bidirectional interactions between users and items:

$$\mathbf{A} = \begin{bmatrix} 0 & \mathbf{R} \\ \mathbf{R}^\top & 0 \end{bmatrix}. \tag{6.6}$$

On such basis, let $\mathcal{G} = \{\mathcal{U} \cup \mathcal{I}, \mathbf{A}\}$ be the user-item bipartite and undirected graph. Moreover, we connote the user- and item-*projected* graphs as $\mathcal{G}_{\mathcal{U}} = \{\mathcal{U}, \mathbf{A}^{\mathcal{U}}\}$ and $\mathcal{G}_{\mathcal{I}} = \{\mathcal{I}, \mathbf{A}^{\mathcal{I}}\}$. Thus, let $\mathbf{R}^{\mathcal{U}}$ and $\mathbf{R}^{\mathcal{I}}$ be the user-user and item-item interaction matrices:

$$\mathbf{R}^{\mathcal{U}} = \mathbf{R} \cdot \mathbf{R}^\top, \qquad \mathbf{R}^{\mathcal{I}} = \mathbf{R}^\top \cdot \mathbf{R}, \tag{6.7}$$

which indicate the co-occurrences among users and items, respectively. Trivially, the adjacency matrices $\mathbf{A}^{\mathcal{U}}$ and $\mathbf{A}^{\mathcal{I}}$ are:

$$\mathbf{A}^{\mathcal{U}} = \mathbf{R}^{\mathcal{U}}, \qquad \mathbf{A}^{\mathcal{I}} = \mathbf{R}^{\mathcal{I}}. \tag{6.8}$$

We use the introduced concepts and notations to describe three topological aspects of the user-item graph and re-interpret them under the lens of recommender systems.

Node degree

By generalizing the definitions of $\mathcal{N}_u$ and $\mathcal{N}_i$, let $\mathcal{N}_u^{(l)}$ and $\mathcal{N}_i^{(l)}$ be the sets of neighborhood nodes for user u and item i at l distance hops. Thus, the node degrees for u and i (i.e., $\sigma_u = |\mathcal{N}_u^{(1)}|$ and $\sigma_i = |\mathcal{N}_i^{(1)}|$) represent the number of item and user nodes directly connected with u and i, respectively. The average user and item node degrees are:

$$\sigma_{\mathcal{U}} = \frac{1}{U} \sum_{u \in \mathcal{U}} |\mathcal{N}_u^{(1)}|, \qquad \sigma_{\mathcal{I}} = \frac{1}{I} \sum_{i \in \mathcal{I}} |\mathcal{N}_i^{(1)}|. \tag{6.9}$$

RecSys re-interpretation. *The node degree in the user-item graph stands for the number of items (users) interacted by a user (item). This is related to the cold-start* *issue in recommendation, where cold users denote low activity on the platform, while cold items are niche products.*

Node degree alone still fails to provide a deeper outlook on the user-item graph. The following topology-aware characteristics, derived from node degree, expand its formulation to other viewpoints.

Clustering

For each partition in a bipartite graph, it is interesting to recognize clusters of nodes in terms of how their neighborhoods overlap, independently of the respective sizes. Let v and w be two nodes from the same partition (e.g., user nodes). Their similarity is the intersection over union of their neighborhoods [168]. By evaluating the metric node-wise, we obtain:

$$\gamma_v = \frac{\sum_{w \in \mathcal{N}_v^{(2)}} \gamma_{v,w}}{|\mathcal{N}_v^{(2)}|}, \qquad \text{with} \quad \gamma_{v,w} = \frac{|\mathcal{N}_v^{(1)} \cap \mathcal{N}_w^{(1)}|}{|\mathcal{N}_v^{(1)} \cup \mathcal{N}_w^{(1)}|}, \tag{6.10}$$

where $\mathcal{N}_v^{(2)}$ is the second-order neighborhood set of v. In this case, we leverage the second-order neighborhood because, in a bipartite graph, nodes from the same partition are connected at (multiple of) 2 hops. The average clustering coefficient on $\mathcal{U}$ and $\mathcal{I}$ is:

$$\gamma_{\mathcal{U}} = \frac{1}{U} \sum_{u \in \mathcal{U}} \gamma_u, \qquad \gamma_{\mathcal{I}} = \frac{1}{I} \sum_{i \in \mathcal{I}} \gamma_i. \tag{6.11}$$

RecSys re-interpretation. *High values of the clustering coefficient indicate that there exists a substantial number of co-occurrences among nodes from the same partition. For instance, when considering the user-side formula, the average clustering coefficient increases if several users share most of their interacted items. The intuition aligns with the rationale behind collaborative filtering: two users are likely to show similar preferences when they interact with the same items.*

The clustering coefficient allows the description of broader portions of the user-item graph compared to the semantics conveyed by node degree. Indeed, the measure takes nodes at 2 hops (i.e., user-item-user and item-user-item connections). Nevertheless, we may want to capture properties for even more extended regions of the graph. For this reason, we introduce one last topology-aware characteristic that goes beyond the 2-hop distance among nodes.

Degree assortativity

In real-world graphs, nodes tend to gather when they share similar characteristics. Such a tendency is measured through the assortativity coefficient. Depending on the semantics of "node similarity", there exist different formulations for assortativity [226]. For the sake of this analysis, we consider the assortativity coefficient based on the scalar properties of graph nodes, for instance, their degree. Let $\mathcal{D} = \{d_1, d_2, \dots\}$ be the set of unique node degrees in the graph, and let e_{d_h,d_k} be the fraction of edges connecting nodes with degrees d_h and d_k. Then, let q_{d_h} be the probability distribution to choose a node with degree d_h after having selected a node with the same degree (i.e., the *excess* degree distribution). The degree assortativity coefficient is calculated as:

$$\rho = \frac{\sum\limits_{d_h,d_k} d_h d_k (e_{d_h,d_k} - q_{d_h} q_{d_k})}{std_q^2}, \tag{6.12}$$

where std_q is the standard deviation of the distribution q. Note that, for its formulation, the degree assortativity is similar to a correlation measure (e.g., Pearson correlation). Following the same rationale of the clustering coefficient, we are interested in finding similarity patterns among nodes from the same partition. For this reason, we first apply the projection of the user-item bipartite graph for both users and items to obtain the user- (i.e., $\mathcal{G}_{\mathcal{U}}$) and item- (i.e., $\mathcal{G}_{\mathcal{I}}$) projected graphs. Then, we calculate the degree assortativity coefficients for $\mathcal{G}_{\mathcal{U}}$ and $\mathcal{G}_{\mathcal{I}}$, namely, $\rho_{\mathcal{U}}$ and $\rho_{\mathcal{I}}$.
RecSys re-interpretation. *In the recommendation scenario, the degree assortativity calculated user- and item-wise is a proxy to represent the tendency of users with the same activity level on the platform and items with the same popularity to gather, respectively. Since we calculate the degree assortativity on the complete user-user and item-item co-occurrence graphs, we deem this characteristic to provide a broader view of the dataset than the clustering coefficient. For this reason, to give an intuition of degree assortativity, we borrow the concept of search space traversal depth in search algorithms theory. That is, we re-interpret degree assortativity in recommendation as a topology-aware characteristic showing a strong look-ahead nature.*

To conclude, we also briefly recall the *classical* characteristics underlying the user-item data as presented in [8, 85].

Space size

The space size estimates the number of all possible interactions that might exist among users and items:

$$\zeta = \sqrt{UI}. \tag{6.13}$$

Shape

The shape of a recommendation dataset is defined as the ratio between the number of users and items:

$$\pi = \frac{U}{I}. \tag{6.14}$$

Density

The density of a recommendation dataset measures the ratio of actual user-item interactions with respect to all possible interactions that might connect all users and items:

$$\delta = \frac{E}{UI}, \tag{6.15}$$

where $E = |\{(u,i) \mid \mathbf{R}_{u,i} = 1\}|$ is the number of interactions existing among users and items in the recommendation data.

Gini coefficient

The Gini coefficient is an estimation of the interactions' concentration for both users and items. When calculated on $\mathcal{U}$ and $\mathcal{I}$, we have:

$$\kappa_{\mathcal{U}} = \frac{\sum_{u=1}^{U-1} \sum_{v=u+1}^{U} abs(\sigma_u - \sigma_v)}{U \sum_{u=1}^{U} \sigma_u}, \qquad \kappa_{\mathcal{I}} = \frac{\sum_{i=1}^{I-1} \sum_{j=i+1}^{I} abs(\sigma_i - \sigma_j)}{I \sum_{i=1}^{I} \sigma_i}, \tag{6.16}$$

where $abs()$ is the function returning the absolute value.

6.3.2 Topological characteristics in graph collaborative filtering

Since graph-based recommender systems are specifically designed to view the user-item interaction data as a bipartite and undirected graph, in this section, we seek to understand how and to what extent such models (explicitly) integrate *topological* data characteristics into their formulations. To this aim, we select four popular and recent approaches in graph collaborative filtering, namely: LightGCN [126], DGCF [328], UltraGCN [217], and SVD-GCN [248].

As additional background with respect to Section 6.3.1, we introduce the notations $\mathbf{e}_u \in \mathbb{R}^b$ and $\mathbf{e}_i \in \mathbb{R}^b$ as the initial embeddings of the nodes for user u and item i, respectively, where $b << U, I$. Then, in the case of message-propagation at different layers, we also introduce the notations $\mathbf{e}_u^{(l)}$ and $\mathbf{e}_i^{(l)}$ to indicate the updated node embeddings for user u and item i after l propagation layers, with $0 \leq l \leq L$ (note that $\mathbf{e}_u^{(0)} = \mathbf{e}_u$ and $\mathbf{e}_i^{(0)} = \mathbf{e}_i$).

LightGCN

He et al. [126] propose to lighten the graph convolutional layer presented in Kipf et al. [158] for the recommendation task. Specifically, their layer removes feature transformation and non-linearities:

$$\mathbf{e}_u^{(l)} = \sum_{i' \in \mathcal{N}_u^{(1)}} \frac{A_{ui'} \mathbf{e}_{i'}^{(l-1)}}{\sqrt{\sigma_u \sigma_{i'}}}, \quad \mathbf{e}_i^{(l)} = \sum_{u' \in \mathcal{N}_i^{(1)}} \frac{A_{iu'} \mathbf{e}_{u'}^{(l-1)}}{\sqrt{\sigma_i \sigma_{u'}}}, \tag{6.17}$$

where each neighbor contribution is weighted through the corresponding entry in the normalized Laplacian adjacency matrix to flatten the differences among nodes with high and low degrees. Since $A_{ui'} = 1, \ \forall i' \in \mathcal{N}_u^{(1)}$ (the dual holds for $A_{iu'}$), the contribution weighting comes only from the denominator.

DGCF

Wang et al. [328] assume that user-item interactions are decomposed into a set of independent intents, representing the specific aspects users may be interested in when interacting with items. In this respect, the authors propose to iteratively learn a set of weighted adjacency matrices $\{\tilde{\mathbf{A}}_1, \tilde{\mathbf{A}}_2, \dots\}$, where each of them records the user-item importance weights based on the specific intent it represents. Then, they introduce a graph disentangling layer for each weighted adjacency matrix:

$$\mathbf{e}_{u,*}^{(l)} = \sum_{i' \in \mathcal{N}_u^{(1)}} \frac{\tilde{A}_{ui',*} \mathbf{e}_{i',*}^{(l-1)}}{\sqrt{\sigma_{u,*} \sigma_{i',*}}}, \quad \mathbf{e}_{i,*}^{(l)} = \sum_{u' \in \mathcal{N}_i^{(1)}} \frac{\tilde{A}_{iu',*} \mathbf{e}_{u',*}^{(l-1)}}{\sqrt{\sigma_{i,*} \sigma_{u',*}}}, \tag{6.18}$$

where $\tilde{A}_{ui',*}$ and $\mathbf{e}_{i',*}^{(l-1)}$ are the learned importance weight of user u on item i' and the embedding of item i' for any intent, while $\sigma_{u,*}$ is the corresponding node degree calculated on $\tilde{\mathbf{A}}_*$ (the same applies for the item side).

UltraGCN

Mao et al. [217] recognize three major limitations in GCN-based message-passing for collaborative filtering, namely, (i) the asymmetric weight assignment to connected nodes when considering user-user and item-item relationships; (ii) the impossibility to diversify the importance of each type of relation (i.e., user-item, user-user, item-item) during the message-passing; (iii) the over-smoothing effect when stacking more than 3 layers. To tackle such issues, the authors propose to go beyond the traditional concept of explicit message-passing, and approximate the infinite-layer message-passing through the following:

$$\mathbf{e}_u = \sum_{i' \in \mathcal{N}_u^{(1)}} \frac{A_{ui'}\sqrt{\sigma_u + 1}\mathbf{e}_{i'}}{\sigma_u \sqrt{\sigma_{i'} + 1}}, \quad \mathbf{e}_i = \sum_{u' \in \mathcal{N}_i^{(1)}} \frac{A_{iu'}\sqrt{\sigma_i + 1}\mathbf{e}_{u'}}{\sigma_i \sqrt{\sigma_{u'} + 1}}. \tag{6.19}$$

Note that the procedure is not repeated for layers $l > 1$, as the method surpasses the concept of iterative message-passing. During the optimization, the model first minimizes a constraint loss that adopts negative sampling to limit the over-smoothing effect:

$$\begin{aligned} \min_{\mathbf{e}_u, \mathbf{e}_i, \mathbf{e}_j} \quad & - \sum_{(u,i) \in \mathbf{R}_s^+} \frac{1}{\sigma_u} \frac{\sqrt{\sigma_u + 1}}{\sqrt{\sigma_i + 1}} log(sig(\mathbf{e}_u^\top \cdot \mathbf{e}_i)) + \\ & - \sum_{(u,j) \in \mathbf{R}_s^-} \frac{1}{\sigma_u} \frac{\sqrt{\sigma_u + 1}}{\sqrt{\sigma_j + 1}} log(sig(-\mathbf{e}_u^\top \cdot \mathbf{e}_j)), \end{aligned} \tag{6.20}$$

where $\mathbf{R}_s^+$ and $\mathbf{R}_s^-$ are pairs of positive and negative interactions sampled from the user-item matrix $\mathbf{R}$, while $log()$ and $sig()$ are the logarithm and sigmoid function. Then, they also take into account the item-projected graph $\mathcal{G}_\mathcal{I}$ and minimize the following:

$$\min_{\mathbf{e}_u, \mathbf{e}_j} - \sum_{(u,i) \in \mathbf{R}_s^+} \sum_{j \in topk(\mathbf{R}_{i,*}^\mathcal{I})} \frac{\mathbf{R}_{i,j}^\mathcal{I}}{\sigma_i^\mathcal{I} - \mathbf{R}_{i,i}^\mathcal{I}} \sqrt{\frac{\sigma_i^\mathcal{I}}{\sigma_j^\mathcal{I}}} log(sig(\mathbf{e}_u^\top \cdot \mathbf{e}_j)), \tag{6.21}$$

where $topk()$ retrieves the top-k values of a matrix row-wise, and $\sigma_*^\mathcal{I}$ is the node degree calculated on $\mathcal{G}_\mathcal{I}$.

SVD-GCN

Peng et al. [248] propose a reformulation of the GCN-based message-passing which leverages the similarities between graph convolutional layers and singular value decomposition (i.e., SVD). Specifically, they rewrite the message-passing introduced in LightGCN by making two aspects explicitly emerge, namely: (i) even- and odd-connection message aggregations, and (ii) singular values and vectors obtained by decomposing the user-item interaction matrix $\mathbf{R}$ through SVD. On such basis, the authors' assumption is that the traditional graph convolutional layer intrinsically learns a low-rank representation of the user-item interaction matrix where components corresponding to larger singular values tend to be enhanced. They reinterpret the over-smoothing effect as an increasing gap between singular values when stacking more and more layers. The embeddings for users and items are obtained as follows:

$$\mathbf{e}_u = \mathbf{p}_u exp(a_1 \boldsymbol{\lambda}) \cdot \mathbf{W}, \qquad \mathbf{e}_i = \mathbf{q}_i exp(a_1 \boldsymbol{\lambda}) \cdot \mathbf{W}, \tag{6.22}$$

where: (i) $\mathbf{p}_u$ and $\mathbf{q}_i$ are the left and right singular vectors of the normalized user-item interaction matrix for user u and item i; (ii) $exp()$ is the exponential function; (iii) a_1 is a tunable hyper-parameter of the model; (iv) $\boldsymbol{\lambda}$ is the vector of the largest singular values of the normalized user-item matrix; (v) $\mathbf{W}$ is a trainable matrix to perform feature transformation. Note that the highest singular value λ_{max} and the maximum node degree $\max(\mathcal{D})$ in the user-item interaction matrix are associated by the following inequality:

$$\lambda_{max} \leq \frac{\max(\mathcal{D})}{\max(\mathcal{D}) + a_2}, \tag{6.23}$$

where a_2 is another tunable hyper-parameter of the model to control the gap among singular values. Similarly to UltraGCN, the authors recognize the importance of different types of relationships during the message-passing (i.e., user-item, user-user, item-item). For this reason, they decide to augment the loss function with other components addressing also the similarities among node embeddings from the same partition:

$$\begin{aligned} \min_{\mathbf{e}_v, \mathbf{e}_w, \mathbf{e}_j} \quad & - \sum_{(v,w) \in (\mathbf{R}_s^*)^+} log(sig(\mathbf{e}_v^\top \cdot \mathbf{e}_w)) \; + \\ & - \sum_{(v,j) \in (\mathbf{R}_s^*)^-} log(sig(-\mathbf{e}_v^\top \cdot \mathbf{e}_j)), \end{aligned} \tag{6.24}$$

where v, w, and j are nodes from the same partition, and $\mathbf{R}^*$ is the interaction matrix of that partition.

To sum up, Table 6.8 shows how such techniques are widely used as baselines in the recent literature and indicates which *topological* characteristics are explicitly involved in their formulations. We observe that:

Table 6.8 Selected models for our study. For each of them, we report year, works using them as baselines, and which *topological* characteristics are integrated in the models' formulations.

Model	**Year**	**Baseline in (2021-)**	**Topological characteristics**
LightGCN [126]	2020	e.g., [182, 254, 343, 349, 371]	**Node degree** used to normalize the adjacency matrix in the message passing.
DGCF [328]	2020	e.g., [96, 179, 329, 331, 388]	**Node degree** used to normalize the adjacency matrix in the message passing.
UltraGCN [217]	2021	e.g., [92, 113, 190, 408, 409]	**Node degree** used for normalization in the infinite layer message passing. The model also learns from the ***item*-projected graph**.
SVD-GCN [248]	2022	e.g., [145, 392]	The formulation for the node embeddings involves the largest singular values of the normalized user-item interaction matrix, whose maximum value is related to the maximum **node degree** of the user-item graph. The model also learns from the ***user*-** and ***item*-projected graphs**.

Observation. *The analyzed graph-based recommender systems explicitly utilize the node degree information during the representation learning phase, each of them in a different way. However, clustering coefficient and degree assortativity, which share similarities with node degree's semantics, do not seem to have an evident representation within the models' formulations. Under this perspective, this analysis will also test what topological aspects graph-based RSs can (un)intentionally capture during their training.*

Indeed, these observations pave the way to a further question: ***are (topological) dataset characteristics influencing the recommendation performance of graph-based recommender systems?***

6.3.3 Proposed analysis

To answer such a question, in the following, we present our proposed analysis to evaluate the impact of *classical* and *topological* characteristics on the performance of graph-based recommender systems. As already done in similar works [8, 85], we decide to design an explanatory statistical model which finds dependencies between dataset characteristics and recommendation performance. In this respect, it becomes imperative to collect a set of samples (with dataset characteristics and the corresponding models' recommendation performance) that is large enough to ensure the statistical significance of the conducted analysis under a certain confidence threshold. Thus, in the following, we first present the approach to generate an extensive set of sub-datasets from three popular recommendation datasets, which represent our set of samples. Then, we specify and justify the design choices for the explanatory model which we adopt in our analysis to fit the dataset characteristics of the sub-datasets to the recommendation performance of graph-based recommender systems measured on the same sub-datasets.

Dataset generation

The literature has recently demonstrated that dataset sampling in collaborative filtering can robustify the training of recommendation models [343], sometimes with ad-hoc solutions performed end-to-end in the downstream task [64, 266]. In this part of the chapter, we propose to adopt such strategies ***for another purpose***, namely, the random generation of synthetic (but meaningful) data to conduct our study. We adopt a similar approach to other studies that examine how *classical* characteristics affect the performance of recommendation models [8, 85]. However, we take it a step further by analyzing these aspects from a *topological* perspective. For this reason, we use node- and edge-dropout strategies to generate the sub-datasets, which have gained recent attention in graph learning literature [281, 343]. Since the goal is to generate sub-datasets exhibiting maximum *topological* diversity, we depart from the conventional method of utilizing a Bernoulli distribution to randomly perturb the adjacency matrix. Instead, we vary the level of aggressiveness (i.e., dropout rate) that involves sampling with node- or edge-dropout. Further details about the two distinct *graph sampling strategies* and *sub-dataset generation* are presented in the following paragraphs.

Graph sampling. Given the bipartite user-item graph $\mathcal{G}$, a dropout rate μ, and a ***`sampling`*** strategy, the algorithm returns the sampled graph $\mathcal{G}_m$. When the ***`sampling`*** strategy is `nodeDropout`, the procedure initially calculates the number of nodes to sample from the graph and uniformly extracts the new set of nodes (i.e., $\mathcal{V}_m$). Then, the adjacency matrix is masked to obtain a new one with the retained nodes only (i.e., $\mathbf{A}_m$). Conversely, when the ***`sampling`*** strategy is `edgeDropout`, the procedure initially calculates the number of edges to sample from the graph and uniformly extracts the new set of edges (i.e., $\mathcal{E}_m$). Then, the adjacency matrix is masked to obtain a new one with the retained edges only (i.e., $\mathbf{A}_m$) and the corresponding set of nodes (i.e., $\mathcal{V}_m$). Finally, in both cases, the graph $\mathcal{G}_m$ is induced through $\mathcal{V}_m$ and $\mathbf{A}_m$.

Sub-dataset generation. The procedure takes the bipartite user-item graph $\mathcal{G}$ and the number of samples M as inputs. Iterating for M times, each sampled graph $\mathcal{G}_m$ is generated (through the previous algorithm) and added to the set of sub-datasets (i.e., $\mathcal{M}$). Specifically, during each iteration, a dropout rate μ is uniformly sampled from the range $[0.7, 0.9]$ to ensure the generation of small sub-samples of the original dataset. Then, either `nodeDropout` or `edgeDropout` is chosen at random, thus the overall procedure is not biased towards one of the ***sampling*** strategies (see also Section 6.3.4). Finally, the graph $\mathcal{G}_m$ is obtained by performing the selected ***sampling*** strategy.

In Algorithm 2 and Algorithm 3 we report the pseudocode to perform graph sampling and the sub-dataset generation, respectively. Note that $uniform_N()$ is the function that uniformly samples N elements from a set, while $uniform()$ samples only one element from a set.

Explanatory model

Statistical models can be utilized to elucidate the relationship between a hypothesized cause of a phenomenon (i.e., independent variables) and its effect (measured through dependent variables). While various potential functions can be used to fit the independent variables to the dependent ones, we opt to utilize a linear regression model for two reasons: (i) to adhere to the same methodology employed in recent studies such as [8, 85], and (ii) to derive explanations on the performance impact of data characteristics through linear dependencies, which represents the most straightforward and intuitive strategy. Following this intuition, we formalize a regression model:

$$\mathbf{y} = \boldsymbol{\epsilon} + \theta_0 + \boldsymbol{\theta}_c \mathbf{X}_c. \tag{6.25}$$

We recall that our goal is to test if the factors related to the data characteristics (i.e., $\mathbf{X}_c$) can explain the effect on the recommendation system's performance (i.e., $\mathbf{y}$). Therefore, in Equation 6.25, we denote by $\boldsymbol{\theta}_c = [\theta_1, \ldots, \theta_C]$ the vector of regression coefficients each of whom is associated with the c-th feature (data characteristic considered here), $\mathbf{X}_c \in \mathbb{R}^{M \times C}$ the matrix containing the data characteristic values for each sample in the training set, and $\mathbf{y}$ the vector containing the values of the performance measure associated with all samples in the training set. Moreover, under the assumption of mean-centered data, θ_0 expresses the expected value of $\mathbf{y}$ (i.e., in this case, the expected recommendation performance). The regression model is trained through Ordinary

Algorithm 2: Graph sampling

```
Input: Bipartite user-item graph $\mathcal{G}$, dropout rate $\mu$, graph sampling strategy
 sampling.
Output: Sampled graph $\mathcal{G}_m$.
if sampling == nodeDropout then
    $N = (U + I) * (1 - \mu)$
    $\mathcal{V}_m \leftarrow uniform_N(\mathcal{U} \cup \mathcal{I})$
    $\mathbf{A}_m \leftarrow mask_{node}(\mathbf{A}, \mathcal{V}_m)$
else if sampling == edgeDropout then
    $N = E * (1 - \mu)$
    $\mathcal{E}_m \leftarrow uniform_N(\mathcal{E}_{u \to i})$
    $\mathbf{A}_m \leftarrow mask_{edge}(\mathbf{A}, \mathcal{E}_m)$
    $\mathcal{V}_m \leftarrow induce(\mathbf{A}_m)$
$\mathcal{G}_m \leftarrow \{\mathcal{V}_m, \mathbf{A}_m\}$
Return $\mathcal{G}_m$.
```

Algorithm 3: Sub-dataset generation.

```
Input: Bipartite user-item graph $\mathcal{G}$, number of samples $M$.
Output: $M$ sampled graphs.
$m \leftarrow 1$
$\mathcal{M} = \{\}$
while $m \leq M$ do
    $\mu \leftarrow uniform([0.7, 0.9])$
    sampling $\leftarrow uniform(\{$nodeDropout, edgeDropout$\})$
    $\mathcal{M} \leftarrow \mathcal{M} \cup sample(\mathcal{G}, \mu,$ sampling$)$
    $m \leftarrow m + 1$
end
Return $\mathcal{M}$.
```

Least Squares (OLS):

$$(\theta_0^*, \boldsymbol{\theta}_c^*) = \min_{\theta_0, \boldsymbol{\theta}_c} \frac{1}{2} \|\mathbf{y} - \theta_0 - \boldsymbol{\theta}_c \mathbf{X}_c\|_2^2 . \tag{6.26}$$

In order to show how the performance of RSs are related to dataset characteristics, we utilize the basic regression model presented in Equation 6.26 with the aim of maximizing the R^2 coefficient. This approach allows us to effectively motivate the impact of the $\boldsymbol{\theta}_c$ coefficients on the recommendation system's effectiveness, as outlined in [104] for any regression model.

6.3.4 Results and discussion

This section aims to test the effectiveness of our proposed explanatory model. Specifically, we seek to answer the following two research questions: **RQ1)** What is the impact of classical and topological characteristics on the performance of graph-based recommender systems?; **RQ2)** Is the generation of sub-datasets through node- and edge-dropout differently influencing the explanations of our model?

Experimental setting

We provide here a detailed description of the experimental settings for our proposed explanatory framework. First, we present the recommendation datasets for this study. Then, we report on the adopted characteristics, along with details about their (optional) value rescaling and denomination. Finally, we describe the methodology we follow to train and evaluate the graph-based recommendation models to foster reproducibility.
Datasets. We use specific versions of Yelp2018 [248], Gowalla [126], and Amazon-Book [327]. The usage of such datasets is motivated by their popularity in graph collaborative filtering [113, 126, 179, 217]. Yelp2018 [25] collects data about users and businesses interactions, Amazon-Book is a sub-category of the Amazon dataset [124], and Gowalla [72] is a social-based dataset where users share their locations. Note that, to provide a coherent calculation of the characteristics, we retained only the subset of nodes and edges for each dataset which induces the widest connected graph. In the following, we present the calculation of characteristics, to experimentally justify them.
Characteristics calculation. Following the same setting as in [85], we generate $M = 600$ sub-datasets from the original ones through the techniques described in Algorithm 2 and Algorithm 3, resulting in a total of 1,800 synthetic samples. Second, inspired by similar works [8, 85], we decide to apply the log10-scale to the formulation of some characteristics to obtain values within comparable order of magnitude, thus making the training of the explanatory model more stable. In Table 6.9 we provide a comprehensive outlook on the set of characteristics, where we apply a renaming scheme for the sake of simple understanding and reference. Furthermore, Table 6.10 displays the statistics of the overall datasets and the aggregated characteristics for the generated samples. Finally, Figure 6.5 empirically supports the usage of the selected characteristics, as they appear loosely correlated.
Reproducibility. We perform the random subsampling strategy to split each sub-dataset into train and test (80% and 20%, respectively). Then, we retain the 10% of the train as validation for the early stopping to avoid overfitting. To train LightGCN,

Table 6.9 Selected *classical* and *topological* characteristics. We report the full name, the symbol, whether it is rescaled via log10, and the shorthand adopted.

Type	**Characteristics**	**Symbol**	**Log10**	**Shorthand**
Classical	Space size	ζ	✓	$SpaceSize_{log}$
	Shape	π	✓	$Shape_{log}$
	Density	δ	✓	$Density_{log}$
	Gini user	$\kappa_{\mathcal{U}}$		*Gini-U*
	Gini item	$\kappa_{\mathcal{I}}$		*Gini-I*
Topological	Average degree user	$\sigma_{\mathcal{U}}$	✓	$AvgDegree\text{-}U_{log}$
	Average degree item	$\sigma_{\mathcal{I}}$	✓	$AvgDegree\text{-}I_{log}$
	Average clustering coefficient user	$\gamma_{\mathcal{U}}$	✓	$AvgClustC\text{-}U_{log}$
	Average clustering coefficient item	$\gamma_{\mathcal{I}}$	✓	$AvgClustC\text{-}I_{log}$
	Degree assortativity user	$\rho_{\mathcal{U}}$		*Assort-U*
	Degree assortativity item	$\rho_{\mathcal{I}}$		*Assort-I*

Table 6.10 Dataset overall statistics and characteristic aggregated statistics (minimum and maximum values, mean, and standard deviation) on the sampled sub-datasets.

	Yelp2018				**Amazon-Book**				**Gowalla**			
	Overall Statistics **Users:** 25,677 **Items:** 25,815 **Interactions:** 696,865				*Overall Statistics* **Users:** 70,679 **Items:** 24,915 **Interactions:** 846,434				*Overall Statistics* **Users:** 29,858 **Items:** 40,981 **Interactions:** 1,027,370			
Characteristics	**Min**	**Max**	**Mean**	**Std**	**Min**	**Max**	**Mean**	**Std**	**Min**	**Max**	**Mean**	**Std**
$SpaceSize_{log}$	0.256	1.393	1.000	0.379	0.405	1.593	1.176	0.384	0.430	1.541	1.161	0.375
$Shape_{log}$	0.019	0.105	0.045	0.014	0.325	0.443	0.407	0.021	-0.149	-0.097	-0.129	0.008
$Density_{log}$	-3.699	-2.693	-3.219	0.358	-3.902	-2.896	-3.497	0.365	-3.889	-2.876	-3.380	0.363
Gini-U	0.443	0.508	0.486	0.008	0.384	0.499	0.459	0.023	0.462	0.512	0.491	0.007
Gini-I	0.500	0.609	0.575	0.019	0.518	0.618	0.586	0.018	0.437	0.502	0.478	0.008
$AvgDegree\text{-}U_{log}$	0.523	0.926	0.758	0.110	0.318	0.609	0.476	0.077	0.603	1.017	0.846	0.115
$AvgDegree\text{-}I_{log}$	0.565	0.955	0.804	0.098	0.682	1.043	0.883	0.096	0.487	0.888	0.717	0.109
$AvgClustC\text{-}U_{log}$	-1.144	-0.662	-0.947	0.126	-0.757	-0.407	-0.602	0.095	-1.211	-0.741	-1.013	0.122
$AvgClustC\text{-}I_{log}$	-1.092	-0.652	-0.922	0.105	-1.124	-0.751	-0.967	0.099	-1.080	-0.614	-0.881	0.124
Assort-U	-0.051	0.235	0.021	0.035	-0.041	0.533	0.052	0.074	0.042	0.544	0.188	0.071
Assort-I	-0.002	0.237	0.067	0.037	0.000	0.842	0.443	0.264	-0.037	0.161	0.021	0.028

DGCF, UltraGCN, and SVD-GCN, we fix their configurations (i.e., hyper-parameters and patience for the early stopping) to the best values according to the original papers, since our scope is not to fine-tune them. Finally, following the literature, we use the Recall@20 calculated on the validation for the early stopping, and evaluate the models by assessing the same metric on the test set. Codes, datasets, and configuration files to reproduce all the experiments are available at this link: https://github.com/sisinflab/Graph-Characteristics.

Impact of characteristics

We assess the impact of *classical* and *topological* characteristics on the accuracy performance (i.e., Recall@20) of graph-based RSs. Table 6.11 displays the results for our proposed explanatory model. In the first row (in light gray) we evaluate the

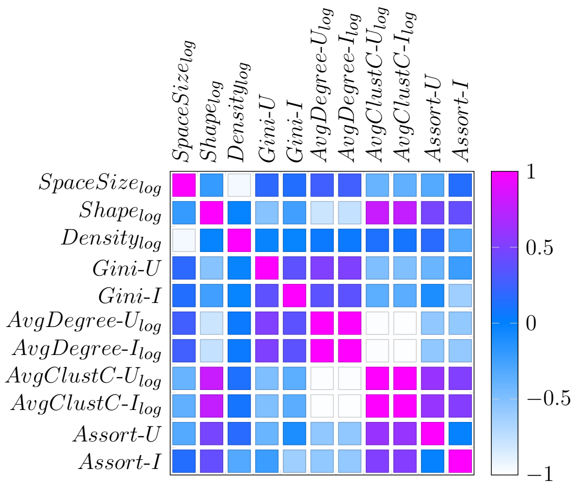


Fig. 6.5 Pearson correlation of the selected characteristics. Many values in $[-0.5, 0.5]$ indicate loosely correlated pairs.

goodness of the explanatory model through the R^2 and its adjusted version denoted as adj. R^2 [8, 85]. Conversely, the remaining rows show the learned characteristics' coefficients (i.e., $[\theta_1, \ldots, \theta_C]$ as described in Section 6.3.3, and renamed through a more human-readable convention). Trivially, coefficients' signs and values indicate whether there exists a (strong) direct/inverse relation between the recommendation metric and the dataset characteristic. Finally, we assess the statistical significance of the results (the asterisks alongside each characteristic's coefficient, refer to the legend below the table).

Overall, the adj. R^2 is, for the vast majority of settings, above 95%, proving the ability of the regression model to explain the accuracy recommendation performance through the measured characteristics. Hereinafter, we further decompose the regression results by categorizing the characteristics as *classical* and *topological*.

Classical dataset characteristics. Previous works [8] have assessed the impact of *classical* characteristics on neighbor- and factorization-based recommendation models. However, a careful search of the relevant literature yields that no study has investigated whether such characteristics influence graph-based recommender systems likewise. From a theoretical standpoint, it could be noticed that neighbor- and factorization-based models have some similarities with graph-based ones, as the latter use a message-passing schema that aggregates information from the *neighborhood* and also learn latent *factor* representations of users and items.

Nevertheless, and interestingly, Table 6.11 suggests that the factorization component might be predominant within graph-based recommender systems. If we refer to the results from [8], we observe that factorization- and graph-based approaches are

Table 6.11 Results of the explanatory model with the Recall@20 as recommendation metric. Besides the row in light gray standing for the R^2, the other rows refer to the learned characteristics' coefficients (with the statistical significance). *Constant* (i.e., θ_0) is the expected value of Recall@20.

Characteristics	**LightGCN**		**DGCF**		**UltraGCN**		**SVD-GCN**	
	Yelp2018	**Gowalla**	**Yelp2018**	**Gowalla**	**Yelp2018**	**Gowalla**	**Yelp2018**	**Gowalla**
R^2(adj. R^2)	0.971(0.971)	0.979(0.978)	0.973(0.973)	0.982(0.981)	0.965(0.964)	0.860(0.858)	0.982(0.981)	0.981(0.981)
Constant	0.100^{***}	0.121^{***}	0.089^{***}	0.107^{***}	0.061^{***}	0.062^{***}	0.116^{***}	0.135^{***}
$SpaceSize_{log}$	0.070^{***}	0.192^{***}	0.133^{***}	0.237^{***}	-0.059^{***}	0.318^{***}	0.064^{***}	0.114^{***}
$Shape_{log}$	-0.253^{*}	-0.231	-0.282^{**}	-0.220^{*}	0.135	-0.003	-0.193	-0.232^{*}
$Density_{log}$	0.194^{***}	0.298^{***}	0.243^{***}	0.327^{***}	0.026^{*}	0.321^{***}	0.203^{***}	0.234^{***}
Gini-U	0.296^{**}	0.104	0.074	-0.071	-0.043	-0.931^{***}	0.136	0.143
Gini-I	1.362^{***}	0.681^{***}	1.108^{***}	0.560^{***}	0.605^{***}	-0.144	1.138^{***}	0.748^{***}
$AvgDegree\text{-}U_{log}$	0.390^{***}	0.605^{***}	0.518^{***}	0.673^{***}	-0.100^{*}	0.640^{***}	0.364^{***}	0.464^{***}
$AvgDegree\text{-}I_{log}$	0.137^{*}	0.374^{***}	0.235^{***}	0.453^{***}	0.034	0.637^{***}	0.171^{**}	0.231^{**}
$AvgClustC\text{-}U_{log}$	0.613^{***}	0.665^{***}	0.726^{***}	0.783^{***}	-0.077	0.706^{***}	0.613^{***}	0.496^{***}
$AvgClustC\text{-}I_{log}$	0.087	0.332^{*}	0.168	0.373^{**}	0.062	0.671^{**}	0.057	0.215
Assort-U	0.094^{***}	0.024^{*}	0.093^{***}	0.013	0.123^{***}	-0.019	0.080^{***}	0.010
Assort-I	-0.051	-0.031	-0.056^{*}	-0.055	0.001	-0.174^{***}	-0.048^{*}	-0.088^{*}

***p-value $\leq$ 0.001, **p-value $\leq$ 0.01, *p-value $\leq$ 0.05

particularly aligned, considering: (i) the inverse correspondence between the accuracy performance metric and the $Shape_{log}$ in almost all settings, meaning that when the number of users is higher than the number of items in the system, accuracy performance may decrease; (ii) the direct correspondence between the accuracy performance metric and $Density_{log}$ and *Gini-I*. As for (ii), the density is historically known as one of the core problems in recommendation (i.e., data sparsity), so it becomes evident why also graph-based recommender systems' performance benefits from denser (i.e., less sparse) datasets. The Gini index measures the dissimilar distribution of items' interactions in the system and could be related to the tendency of RSs to promote popular items from the catalog. That is, when there exist items that have been experienced more frequently than others, both graph- and factorization-based RSs may be biased to popular items, and so their accuracy performance increases. Noteworthy, all such observations are supported by the statistical significance of the results.

Topological dataset characteristics. Graph-based recommendation systems interpret the user-item interaction data as a bipartite and undirected graph. Consequently, this study assesses the influence of *topological* characteristics of the selected recommendation datasets on the accuracy performance.

The most evident outcome is that $AvgDegree\text{-}U_{log}$ and $AvgDegree\text{-}I_{log}$ show a direct correspondence with recommendation accuracy performance in almost all settings. Indeed, this analytically confirms what we already observed in Section 6.3.2 regarding the explicit presence of the node degree in the formulations of all the selected graph-based approaches. In practical terms, when graph-based models are trained on datasets with several interactions for users and items, they learn accurate users' preferences since each node receives the contribution of numerous neighbor nodes. It is worth noticing that, in absolute values, *AvgDegree-*U_{log} is more influential than *AvgDegree-*I_{log} on the overall performance. Hence, under the same average degree gain, an improvement in the user average degree is preferable since it would more significantly improve the overall performance.

As far as clustering coefficient and degree assortativity are concerned, we assess how similarities among nodes from the same partition in the graph may impact the recommendation accuracy performance of models. In terms of *AvgClustC-*U_{log} and *AvgClustC-*I_{log}, the results prove again a strong direct correspondence in almost all settings of graph-based models and datasets. Differently from the average degree scenario, the relative importance of the user-side values is much higher than the one of the item-side for LightGCN and DGCF, while the gap sometimes gets narrower in the case of UltraGCN and SVD-GCN. This may be because while LightGCN and DGCF only leverage user-item types of interactions, UltraGCN and (especially) SVD-GCN also embed the information conveyed in the user- and item-projected graphs in their formulations, thus flattening the different influence of the user-side characteristics over the item-side counterpart.

Interestingly, the *Assort-U* and *Assort-I* characteristics exhibit a direct and inverse correspondence to the accuracy metric, respectively. Furthermore, models such as LightGCN and DGCF have slightly larger coefficients for both *Assort-U* and *Assort-I* than SVD-GCN. Again, these results have a mathematical justification. Indeed, the strong *lookahead* nature of the assortativity measures (refer again to Section 6.3.1) seems to be captured by the multi-layer message-passing performed by LightGCN and DGCF. Conversely, in the case of SVD-GCN, they are less influential, probably because the model acts on the singular values of the adjacency matrix with the effect of limiting the graph convolutional layers' depth to avoid over-smoothing. However, it is important to observe that the assortativity results are less statistically-significant than the others, so we plan to further investigate this aspect in future work. On the contrary, a different trend can be observed for UltraGCN, where both *Assort-U* and *Assort-I* generally have bigger coefficients with much more statistically significant values. Again, this behavior could have a theoretical foundation since the model adopts the infinite-layer approximation, which (differently from SVD-GCN) may capture long-distance relationships in the user-item graph.

Summary. *The analytical and theoretical observations show that: (i) factorization-based approaches could be the core component of graph-based recommender systems; (ii) while confirming its influence on the recommendation performance, node degree* *seems not to be a key topological characteristic to distinguish among the different graph-based models; indeed, the wider perspective provided by clustering coefficient and (especially) degree assortativity may help to recognize how the different models address the topological properties of the graph, even with unexpected outcomes.*

Influence of node- and edge-dropout

The current section investigates the influence of the different sampling strategies, namely, node- and edge-dropout, on the explanatory model. Given the lack of space, we report an extensive analysis of the largest dataset, Gowalla, by considering the performance of LightGCN and SVD-GCN.

As generally observed in real-world networks, user-item bipartite graphs follow the typical trend of *scale-free* networks [255]. Thus, we introduce the following statement on the possible influence of node- and edge-dropout on our linear explanatory model:
Statement. *In general, the node-dropout strategy drops wider portions of the original topology than the edge-dropout, so it may negatively impact the significance of the linear model explanations.*

Figure 6.6 displays the relation (i.e., the black points) between the probability distribution of node degrees in the original graph and their degree values on the Gowalla dataset. As evident, high-degree nodes are less popular than low-degree ones, and this resembles the tendency of real-world networks to be *scale-free* [255]. To be more precise, the actual degree probability distribution approximates neither the *power-law* (i.e., representing *scale-free* networks, in green), nor the *exponential* function (i.e., in red), but it would be well-approximated by a function in-between. This suggests that the high-degree nodes are even less frequent than they usually are in *scale-free* networks.

The figure helps to re-interpret the impact of node- and edge-dropout. While node-dropout works by removing nodes (and all the edges connected to them), edge-dropout eliminates edges and the consequently-disconnected nodes. Let us consider their worst-case scenarios. For node-dropout, it would be to drop many high-degree nodes from the graph. Whereas, when considering edge-dropout, it would be to drop all the edges connected to several nodes and thus disconnect them from the graph.

This intuition drove us towards stating that, on averagely, node-dropout has the potential to drop larger portions of the user-item graph than edge-dropout. Indeed, this might undermine the goodness of the explanations produced by our explanatory framework. The assumption further motivates the strategy we adopted to generate the sub-datasets in RQ1, where we performed both node- and edge-dropout by uniformly

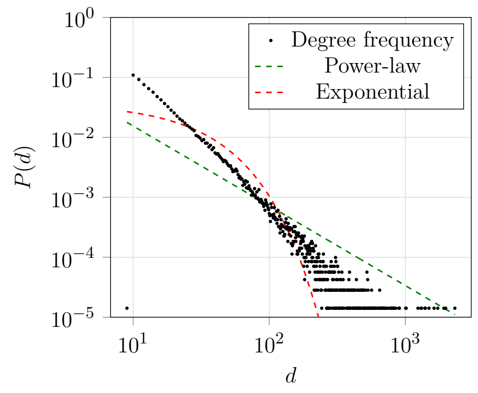


Fig. 6.6 Node degree probability distribution on Gowalla. The black points (i.e., the real data) would be approximated by a function in-between the *power-law* and the *exponential*.

selecting one of them for each sampled sub-dataset in order not to bias the procedure towards either node- or edge-dropout (see again Section 6.3.3).

To analytically test the statement, we build four versions of the dataset $\mathbf{X}_c$, each with varying portions of sub-datasets generated through node and edge dropout, respectively (refer to Equation 6.25). Specifically, the number of samples in $\mathbf{X}_c$ changes in accordance to:

$$|\mathbf{X}_c| = (1-\alpha)|\mathbf{X}_c^n| + \alpha|\mathbf{X}_c^e|, \tag{6.27}$$

where $\mathbf{X}_c^n$ and $\mathbf{X}_c^e$ indicate the portion of $\mathbf{X}_c$ sampled through node- and edge-dropout, while α is a parameter to control the number of samples from $\mathbf{X}_c^n$ and $\mathbf{X}_c^e$ contributing to the final dataset $\mathbf{X}_c$. We use $|\cdot|$ as a necessary notation abuse to refer to any dataset size in a simple way. We let α range in $\{0.0, 0.3, 0.7, 1.0\}$, where extreme values of α are used to build the dataset through either node- or edge-dropout; the others combine the two sampling strategies.

Table 6.12 reports the explanatory model results for the Recall@20 as accuracy metric at varying α values, along with the average sampling statistics on each setting of α. In alignment with the above statement, the average sampling statistics show that node-dropout generally retains smaller portions of the graph than the edge-dropout. Then, the regression results highlight that the optimal trade-off between high R^2 (adj. R^2) and statistical significance of the learned coefficients is reached when combining samples generated through both node- and edge-dropout. On the contrary, the settings with either node- or edge-dropout do not offer the conditions for the regression model to learn meaningful dependencies, with respect to the R^2 (adj. R^2) and/or statistical significance. Indeed, in the extreme cases, the characteristic-performance dependencies are not aligned with the ones observed in RQ1 either on the sign or on the absolute value of the coefficients. This justifies the dataset sampling adopted to explore RQ1.

Table 6.12 Results of the explanatory model on Gowalla (Recall@20) obtained with LightGCN and SVD-GCN, for different proportions of sub-datasets generated through node- and edge-dropout. The header reports a graphical intuition of α's variation and average sampling statistics.

	Node drop ●●●	**Edge drop ○○○**	**Node drop ●●○**	**Edge drop ●○○**	**Node drop ●○○**	**Edge drop ●●○**	**Node drop ○○○**	**Edge drop ●●●**
	Average Sampling Statistics **Users:** 5,828 **Items:** 7,887 **Interactions:** 45,620		*Average Sampling Statistics* **Users:** 12,744 **Items:** 17,229 **Interactions:** 97,785		*Average Sampling Statistics* **Users:** 21,730 **Items:** 29,316 **Interactions:** 160,919		*Average Sampling Statistics* **Users:** 28,526 **Items:** 38,467 **Interactions:** 209,659	
Characteristics	**LightGCN**	**SVD-GCN**	**LightGCN**	**SVD-GCN**	**LightGCN**	**SVD-GCN**	**LightGCN**	**SVD-GCN**
R^2(adj. R^2)	0.597(0.583)	0.754(0.745)	0.968(0.967)	0.970(0.969)	0.986(0.985)	0.987(0.987)	0.994(0.994)	0.991(0.991)
Constant	0.179***	0.193***	0.146***	0.159***	0.098***	0.112***	0.062***	0.077***
$SpaceSize_{log}$	0.092	0.037	0.143***	0.064**	0.220***	0.136***	−0.162***	−0.048
$Shape_{log}$	−0.078	−0.118	−0.265	−0.261*	−0.175	−0.303	0.084	−0.157
$Density_{log}$	−0.079**	−0.090**	0.261***	0.195***	0.323***	0.251***	0.072**	0.040
Gini-U	−0.013	0.015	0.184	0.193	0.246	0.224	0.291***	0.225*
Gini-I	0.883***	0.884***	0.856***	0.867***	0.911***	0.940***	0.389***	0.387***
$AvgDegree\text{-}U_{log}$	0.052	0.005	0.536***	0.390***	0.631***	0.539***	−0.132*	0.070
$AvgDegree\text{-}I_{log}$	−0.026	−0.112	0.271**	0.129	0.456***	0.236*	−0.048	−0.087
$AvgClustC\text{-}U_{log}$	0.209	0.168	0.654***	0.508**	0.687**	0.647***	−0.133	0.016
$AvgClustC\text{-}I_{log}$	−0.141	−0.227	0.137	−0.007	0.436	0.172	−0.145	−0.112
Assort-U	0.008	−0.001	0.017	0.008	0.013	−0.002	0.011	−0.003
Assort-I	−0.022	−0.078**	0.028	−0.057	0.059	−0.056	0.012	−0.037

****p-value* $\leq$ *0.001, **p-value* $\leq$ *0.01, *p-value* $\leq$ *0.05*

Table 6.13 Additional results for RQ1 on Amazon-Book. The current table is to be interpreted the same way as Table 6.11.

Characteristics	**LightGCN**	**DGCF**	**UltraGCN**	**SVD-GCN**
R^2(adj. R^2)	0.953(0.952)	0.951(0.950)	0.800(0.797)	0.964(0.963)
Constant	0.088***	0.079***	0.056***	0.112***
$SpaceSize_{log}$	0.456***	0.364***	0.067	0.433***
$Shape_{log}$	−0.668***	−0.598***	−0.215	−0.582***
$Density_{log}$	0.546***	0.453***	0.141**	0.541***
Gini-U	−0.073	−0.085	−0.350**	−0.042
Gini-I	1.302***	1.148***	0.772***	1.306***
$AvgDegree\text{-}U_{log}$	1.336***	1.116***	0.316*	1.265***
$AvgDegree\text{-}I_{log}$	0.668***	0.518***	0.101	0.683***
$AvgClustC\text{-}U_{log}$	1.627***	1.307***	0.210	1.617***
$AvgClustC\text{-}I_{log}$	0.431***	0.337***	0.111	0.354***
Assort-U	0.031	0.041*	0.070***	0.028
Assort-I	−0.010	−0.004	0.025***	−0.010*

****p-value* $\leq$ *0.001, **p-value* $\leq$ *0.01, *p-value* $\leq$ *0.05*

Summary. *The empirical and analytical evaluation of the explanatory model for different settings of node- and edge-dropout indicates that their simultaneous combination to generate the sub-datasets (i.e., the strategy we followed in RQ1) is beneficial to produce meaningful explanations.*

The interested reader may refer to Tables 6.13 and 6.14 for additional results of RQ1 and RQ2.

6.4 How neighborhood exploration influences novelty and diversity

To mitigate the over-smoothing effect, graph-based techniques for collaborative filtering limit the exploration of neighborhood to three hops [61, 126, 325]. Similar approaches

Table 6.14 Additional results for RQ2 on DGCF and UltraGCN. The current table is to be interpreted the same way as Table 6.12.

	Node drop ●●●	Edge drop ○○○	Node drop ●●○	Edge drop ●○○	Node drop ●○○	Edge drop ●●○	Node drop ○○○	Edge drop ●●●
	Average Sampling Statistics **Users:** 5,828 **Items:** 7,887 **Interactions:** 45,620		*Average Sampling Statistics* **Users:** 12,744 **Items:** 17,229 **Interactions:** 97,785		*Average Sampling Statistics* **Users:** 21,730 **Items:** 29,316 **Interactions:** 160,919		*Average Sampling Statistics* **Users:** 28,526 **Items:** 38,467 **Interactions:** 209,659	
Characteristics	**DGCF**	**UltraGCN**	**DGCF**	**UltraGCN**	**DGCF**	**DGCF**	**UltraGCN**	
R^2(adj. R^2)	0.888(0.884)	0.597(0.583)	0.973(0.972)	0.833(0.827)	0.988(0.988)	0.883(0.879)	0.994(0.994)	0.599(0.584)
Constant	0.162***	0.091***	0.131***	0.074***	0.085***	0.051***	0.051***	0.034***
$SpaceSize_{log}$	0.130**	0.175*	0.192***	0.358***	0.264***	0.337***	−0.188***	1.644***
$Shape_{log}$	−0.109	−0.096	−0.232*	−0.022	−0.136	−0.029	−0.011	1.055**
$Density_{log}$	−0.005	0.394***	0.294***	0.355***	0.351***	0.340***	0.168***	0.185
Gini-U	−0.136	−0.824***	0.083	−0.880***	0.130	−0.775**	0.200*	−0.961
Gini-I	0.756***	−0.152	0.717***	−0.248	0.668***	−0.333	0.264**	−0.732
$AvgDegree\text{-}U_{log}$	0.179**	0.617***	0.601***	0.723***	0.684***	0.692***	−0.014	1.302***
$AvgDegree\text{-}I_{log}$	0.070	0.521**	0.369***	0.702***	0.547***	0.663***	−0.025	2.356***
$AvgClustC\text{-}U_{log}$	0.362***	0.619**	0.715***	0.641**	0.706***	0.481	0.001	1.099**
$AvgClustC\text{-}I_{log}$	−0.080	0.412	0.252	0.903**	0.571**	0.962**	−0.083	2.415***
Assort-U	−0.001	0.001	0.011	0.000	0.001	0.012	0.008	0.064
Assort-I	−0.050	−0.111**	0.002	−0.151**	0.002	−0.086	−0.032	−0.141

***p-value $\leq$ 0.001, **p-value $\leq$ 0.01, *p-value $\leq$ 0.05

are designed to weight the importance of each neighbor node on its ego node through attention mechanisms [313], which allows the exploration of even smaller portions of the neighborhood to reach remarkable results [328].

Conversely, recent works [217, 278] highlight critical limitations in the adoption of graph convolution to explore users' and items' neighborhoods. Starting from the idea described in [126], they propose alternative reformulations of GCN for the recommendation task, providing simplified and lighter versions which go beyond the traditional concept of multi-hop message-passing. By comparing these latter approaches to the ones described earlier, we might categorize them all into two families, namely, graph recommendation techniques performing **`explicit`** (e.g., [61, 126, 325, 328]) and **`implicit`** (e.g., [217, 278]) message-passing.

Although the literature has widely shown the recommendation accuracy boost of such models to traditional (i.e., non-graph) CF baselines, their ability to produce *novel* and *diverse* recommendation lists [310, 311] remains poorly investigated. While the topic of *multi-objective* recommendation has been addressed only recently by few works in graph CF [292, 401], modern recommender systems are more and more required to reach a sufficient trade-off between accurate and novel/diverse recommendations [167, 287, 360], as a renewed need from both user's and business's perspectives [2, 5, 166].

This fourth part of the chapter seeks to understand how and why the *neighborhood exploration* strategy and (optionally) *depth* may influence novelty and diversity recommendation metrics in graph collaborative filtering. To this aim, we run extensive experiments by training and evaluating six state-of-the-art graph CF models on three popular recommendation datasets.

Our contributions are threefold: (i) to the best of our knowledge, no previous work has evaluated approaches from the two recognized graph recommendation families (i.e., **`explicit`** and **`implicit`** message-passing) on a grid of accuracy/novelty/diversity recommendation metrics, (ii) to provide a fair comparison, we train all **`explicit`** message-passing models exploring the whole hop range 1-4, which also allows examining the accuracy/novelty/diversity trade-off on the neighborhood size, and (iii) we propose a simple reformulation of the **`explicit`** message-passing schema where **`same`**-type node connections (e.g., user-user) and **`different`**-type node connections (e.g., user-item) are formally highlighted, in an effort to unveil their influence on the metrics' trade-off.

6.4.1 Novelty and diversity in recommendation

User experience is becoming crucial on recommendation platforms [141, 149, 282] as the suggestion of interesting lists of items satisfies users and entices them to remain loyal to the platform, thus increasing profits [316]. A good user experience requires the recommended items to be nontrivial, as diverse as possible, and possibly unexpected [110, 282]. However, designing dedicated models is particularly challenging due to the inherent difficulty of evaluating them without a user study. For this reason, researchers have dedicated a considerable effort to the beyond-accuracy dimensions over the past two decades [276, 311, 379]. While the search for the accuracy/novelty/diversity trade-off has gained momentum in recommendation [5, 15, 167, 287, 360], to the best of our knowledge, only two studies investigate novelty and diversity dimensions in the field of graph collaborative filtering [292, 401]. They focus on identifying the accuracy/diversity trade-off by proposing specific models that could achieve competitive performance. However, they do not deepen into analyzing the influence of *neighborhood exploration* on the highlighted dimensions.
On the contrary, we assess the state-of-the-art, most accurate models for graph recommendation and inspect how they behave on novelty and diversity, exploring the potential motivations with a focus on their different neighborhood exploration strategies.

6.4.2 Reformulating explicit message-passing

Starting from the novel model classification for graph collaborative filtering outlined here (i.e., neighborhood exploration approaches leveraging **`explicit`** or **`implicit`** message-passing), in this section we propose a simple (but useful) reformulation for the former family where **`same`**- and **`different`**-type node interactions (e.g., user-user and user-item, respectively) are formally highlighted.

Preliminaries

Let $\mathcal{U} = \{u_1, u_2, \dots, u_N\}$ and $\mathcal{I} = \{i_1, i_2, \dots, i_M\}$ be the sets of users and items. Starting from $\mathcal{U}$ and $\mathcal{I}$, we consider the bipartite and undirected graph connecting pairs of nodes (i.e., users and items) with an existing interaction among them. User and item node features are the embeddings $\mathbf{e}_u \in \mathbb{R}^d, \forall u \in \mathcal{U}$ and $\mathbf{e}_i \in \mathbb{R}^d, \forall i \in \mathcal{I}$, respectively.

Traditional message-passing

Let u and i be the nodes for the user and the item to update (*ego* nodes), and let $\mathcal{N}(u)$ and $\mathcal{N}(i)$ be the sets of nodes at one hop from u and i, respectively (*neighbor* nodes). The schema aggregates the embeddings from the neighborhood (*messages*) to refine the ego nodes:

$$\mathbf{e}_u^{(1)} = \omega\left(\left\{\mathbf{e}_{i'}^{(0)}, \forall i' \in \mathcal{N}(u)\right\}\right), \quad \mathbf{e}_i^{(1)} = \omega\left(\left\{\mathbf{e}_{u'}^{(0)}, \forall u' \in \mathcal{N}(i)\right\}\right) \tag{6.28}$$

where $\mathbf{e}_u^{(1)}$ and $\mathbf{e}_i^{(1)}$ are the refined embedding versions of user u and item i after one hop, $\omega(\cdot)$ is the aggregation function (e.g., the summation), while $\mathbf{e}_{u'}^{(0)} = \mathbf{e}_{u'}$ and $\mathbf{e}_{i'}^{(0)} = \mathbf{e}_{i'}$. To explore deeper and deeper neighborhoods of the ego nodes, aggregation is usually iterated. After two hops, the embeddings of user u and item i are:

$$\mathbf{e}_u^{(2)} = \omega\left(\left\{\mathbf{e}_{i'}^{(1)}, \forall i' \in \mathcal{N}(u)\right\}\right), \quad \mathbf{e}_i^{(2)} = \omega\left(\left\{\mathbf{e}_{u'}^{(1)}, \forall u' \in \mathcal{N}(i)\right\}\right) \tag{6.29}$$

Thus, the general message-passing formulation after l hops is:

$$\mathbf{e}_u^{(l)} = \omega\left(\left\{\mathbf{e}_{i'}^{(l-1)}, \forall i' \in \mathcal{N}(u)\right\}\right), \quad \mathbf{e}_i^{(l)} = \omega\left(\left\{\mathbf{e}_{u'}^{(l-1)}, \forall u' \in \mathcal{N}(i)\right\}\right) \tag{6.30}$$

Proposed reformulation

The two-hop node update in Equation (6.29) is further expanded through the one-hop node update in Equation (6.28):

$$\begin{aligned}
\mathbf{e}_u^{(2)} &= \omega\Bigg(\Bigg\{\omega\Bigg(\Bigg\{\mathbf{e}_{u''}^{(0)}, \underbrace{\forall u'' \in \mathcal{N}(i') \setminus \{u\}}_{\text{2-hop}}\Bigg\}\Bigg), \underbrace{\forall i' \in \mathcal{N}(u)}_{\text{1-hop}}\Bigg\}\Bigg) \\
\mathbf{e}_i^{(2)} &= \omega\Bigg(\Bigg\{\omega\Bigg(\Bigg\{\mathbf{e}_{i''}^{(0)}, \underbrace{\forall i'' \in \mathcal{N}(u') \setminus \{i\}}_{\text{2-hop}}\Bigg\}\Bigg), \underbrace{\forall u' \in \mathcal{N}(i)}_{\text{1-hop}}\Bigg\}\Bigg)
\end{aligned} \tag{6.31}$$

where set differences are used to avoid node duplicates. After two hops, the node embeddings of user u and item i get the contributions of those users u'' and items i'' for

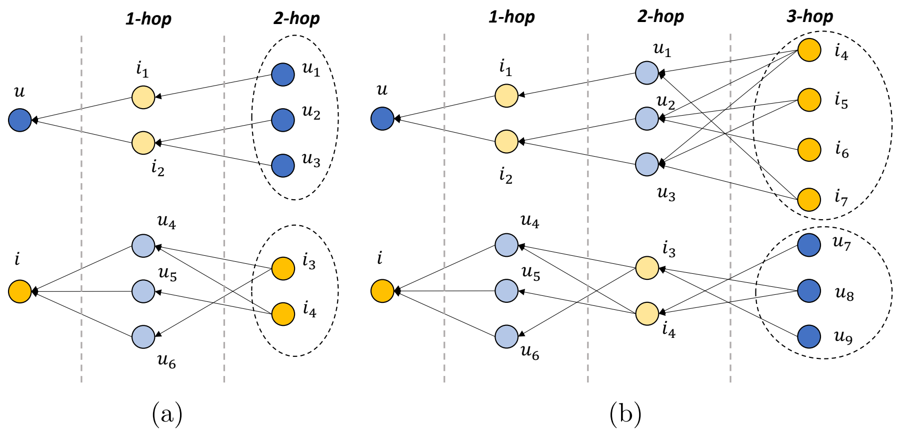


Fig. 6.7 User and item neighborhood exploration after (a) 2 and (b) 3 hops. Contributions to the ego node update are highlighted through dashed ovals. Edge direction indicates the message propagation from neighbor to ego nodes.

whom there exists a *user-item-user* path connecting u with u'', and an *item-user-item* path connecting i with i'', respectively (Figure 6.7a). Such paths link **`same`**-type nodes. In a similar manner, let us apply the general formula from Equation (6.30) to the three-hop node update:

$$\mathbf{e}_u^{(3)} = \omega\left(\left\{\mathbf{e}_{i'}^{(2)}, \forall i' \in \mathcal{N}(u)\right\}\right), \quad \mathbf{e}_i^{(3)} = \omega\left(\left\{\mathbf{e}_{u'}^{(2)}, \forall u' \in \mathcal{N}(i)\right\}\right) \tag{6.32}$$

which we expand through Equation (6.31):

$$\begin{aligned} \mathbf{e}_u^{(3)} &= \omega\Bigg(\Bigg\{\omega\Bigg(\Bigg\{\omega\Bigg(\Bigg\{\mathbf{e}_{i'''}^{(0)}, \underbrace{\forall i''' \in \mathcal{N}(u'') \setminus \{i''\}}_{\text{3-hop}}\Bigg\}\Bigg), \\ &\underbrace{\forall u'' \in \mathcal{N}(i') \setminus \{u''\}}_{\text{2-hop}}\Bigg\}\Bigg), \underbrace{\forall i' \in \mathcal{N}(u)}_{\text{1-hop}}\Bigg\}\Bigg) \\ \mathbf{e}_i^{(3)} &= \omega\Bigg(\Bigg\{\omega\Bigg(\Bigg\{\omega\Bigg(\Bigg\{\mathbf{e}_{u'''}^{(0)}, \underbrace{\forall u''' \in \mathcal{N}(i'') \setminus \{u''\}}_{\text{3-hop}}\Bigg\}\Bigg), \\ &\underbrace{\forall i'' \in \mathcal{N}(u') \setminus \{i''\}}_{\text{2-hop}}\Bigg\}\Bigg), \underbrace{\forall u' \in \mathcal{N}(i)}_{\text{1-hop}}\Bigg\}\Bigg) \end{aligned} \tag{6.33}$$

After three hops, the node embeddings of user u and item i get the contributions of those items i''' and users u''' for whom there exists a *user-item-user-item* path connecting u with i''', and an *item-user-item-user* path connecting i with u''', respectively (Figure 6.7b). In this case, such paths link **`different`**-type nodes.

This reformulation outlines two neighborhood exploration types, propagating messages through **`same`**- and **`different`**-type nodes after an even and an odd number of hops, respectively. While previous works assess recommendation performance when indistinctly increasing the hop numbers, we provide a finer evaluation based on the type of the explored nodes. In the next sections, we will count hops following the introduced categorization. For example, **`same`**-type node explorations after 1 and 2 hops refer to the paths *user-item-user* and *user-item-user-item-user*, respectively, while **`different`**-type node explorations after 1 and 2 hops refer to the paths *user-item* and *user-item-user-item*, respectively.

Before deepening into the presentation and discussion of the experiments and results, it is worth clarifying that what we propose here does not represent a way to recast GNNs-based recommendation approaches under the relational-GNNs [273] perspective. Indeed, the GNN architectures considered in recommendation work on graphs with undirected edges which stand for single-typed links (i.e., "user has interacted with" and viceversa); thus, when we refer to specific paths such as *user-item-user-item*, we are actually considering a *multi-hop* exploration (consisting of multiple "user has interacted with" kind of links) of the neighborhood. Bringing this clarification and what has been said above together, we may state our proposed reformulation has the main and only purpose to reconsider message-passing in GNNs approaches for recommendation as a "typed" message-passing, meaning that the message-passing after an *even* and *odd* number of explorations is connecting **same**- and **different**-type nodes in the graph, respectively; however, the most (and only) atomic relation in the graph still remains of one type ("user has interacted with").

6.4.3 Experiments and discussion

In the following, we describe datasets, baselines, reproducibility details, evaluation protocol, and results of our analysis.

Experimental setup

Datasets. We adopt Movielens-1M [121], Amazon Digital Music [218], and Epinions [261]. Following a similar approach to [14], these datasets are binarized by retaining interactions with a score greater than 3 (Epinions already has an implicit version) and filtered through the p-core to avoid the cold-start effect [124, 125] which is out of the scope of this analysis. Movielens-1M counts 5,915 users, 2,753 items, and 570,622 interactions, Amazon Digital Music counts 8,328 users, 6,275 items, and 99,400 interactions, and Epinions counts 14,341 users, 13,145 items, and 269,170 interactions. All datasets statistics are fully reported in Table 6.15.

Baselines. We evaluate graph recommendation models adopting `explicit` and `implicit` message propagation.

`Explicit` *message-passing*

- **Neural graph collaborative filtering (NGCF) [325]** proposes to refine users' and items' collaborative embeddings by using a GCN-like model which explores the neighborhood and the inter-dependencies among ego and neighbor nodes.
- **Light graph convolutional network (LightGCN) [126]** lightens and improves the NGCF architecture by removing the embedding projections and non-linear activations in each propagation layer.
- **Disentangled graph collaborative filtering (DGCF) [328]** weights the importance of neighbor nodes on the ego node by disentangling the intents involved in each user/item interaction for the sake of explainability.
- **Linear residual graph convolutional collaborative filtering (LR-GCCF) [61]** improves the LightGCN approach by introducing a novel residual block in the convolutional layer for the user-item preference prediction.

`Implicit` *message-passing*

- **Ultra simplification of graph convolutional networks (UltraGCN) [217]** introduces additional objective function components to approximate infinite propagation layers and learn useful item-item connections.
- **Graph filter based collaborative filtering (GFCF) [278]** leverages graph signal processing to formulate a closed-form user-item preference prediction based upon the bipartite graph.

Reproducibility. Datasets are split into train/validation/test with the 80/10/10 hold-out. Models are trained by searching the best hyperparameters as in [33] and setting search spaces according to the original works while fixing the number of epochs to 400 and batch size to 1024. Our implementation is based upon the Elliot framework for reproducible recommender systems [11]. To foster the future reproduction of this analysis, datasets, codes, and configuration files are made accessible[16].

[16]https://github.com/sisinflab/Novelty-Diversity-Graph.

Table 6.15 Statistics of the tested datasets.

Datasets	**#Users**	**#Items**	**#Interactions**	**Sparsity**
Movielens-1M	5,915	2,753	570,622	0.9650
Amazon Digital Music	8,328	6,275	99,400	0.9981
Epinions	14,341	13,145	269,170	0.9986

Evaluation. First, we use the recall (Recall@k) and the normalized discounted cumulative gain (nDCG@k) to measure the recommendation **Accuracy** of the baselines. Then, following [310, 311], we select the expected popularity complement (EPC@k) and the expected free discovery (EFD@k) as **Novelty** metrics [311], along with the 1's complement of the Gini index (Gini@k) and the Shannon entropy (SE@k) as **Diversity** metrics [276]. Both the EPC@k and the EFD@k account for long-tail items and measure the expected number of recommended unknown and known items, which are also relevant, respectively. The Gini@k and the SE@k calculate how unequally a recommender system shows different items to users. We set the Recall@20 as validation metric to follow the original papers. For each recommendation metric, higher values stand for better performance.

Results and discussion

This section shows the recommendation performance of the tested baselines from a general and a finer evaluation of the accuracy/novelty/diversity trade-offs. All reported results refer to the top-20 recommendation lists.

Overall recommendation performance. Table 6.16 depicts recommendation performance on accuracy, novelty, and diversity, when comparing **`explicit`** to **`implicit`** message-passing graph approaches in their best configuration.

Coherently with the literature, DGCF and LR-GCCF are steadily the best or the second-to-best models on accuracy (e.g., DGCF reaches the second-to-best Recall on Amazon Digital Music, while LR-GCCF obtains the best nDCG on Movielens-1M). Approaches with **`implicit`** message aggregation (i.e., UltraGCN and GFCF) still compete with the other baselines on accuracy (e.g., GFCF is the best model on Amazon Digital Music for the Recall and the nDCG, and UltraGCN is the best technique on Epinions for the nDCG).

As for the accuracy/novelty/diversity trade-off, we see that, independently of the adoption of message-passing, accurate approaches can also produce novel recommendations (e.g., LR-GCCF and DGCF are the best and second-to-best approaches for accuracy and novelty on Movielens-1M, and GFCF and UltraGCN provide superior

Table 6.16 Overall recommendation performance on accuracy, novelty, and diversity metrics for top-20 recommendation lists, when comparing **`explicit`** to **`implicit`** message propagation. Bold and underline stand for best and second-to-best values, respectively.

Models	**Movielens-1M**						**Amazon Digital Music**						**Epinions**					
	Accuracy		**Novelty**		**Diversity**		**Accuracy**		**Novelty**		**Diversity**		**Accuracy**		**Novelty**		**Diversity**	
	Recall	nDCG	EPC	EFD	Gini	SE	Recall	nDCG	EPC	EFD	Gini	SE	Recall	nDCG	EPC	EFD	Gini	SE
MostPop	0.1380	0.1099	0.0473	0.5365	0.0105	5.2156	0.0319	0.0154	0.0029	0.0263	0.0031	4.3832	0.0467	0.0224	0.0054	0.0489	0.0015	4.4358
Random	0.0077	0.0060	0.0036	0.0414	0.9105	11.4085	0.0017	0.0007	0.0002	0.0021	0.8929	12.5890	0.0015	0.0006	0.0002	0.0024	0.8789	13.6486
`Explicit` message-passing																		
NGCF	0.2535	0.1985	0.0929	1.0214	**0.1479**	8.9930	0.1127	0.0606	0.0109	0.1270	**0.4130**	**11.6953**	0.0792	0.0394	0.0096	0.1079	**0.2107**	**11.6255**
LightGCN	0.2712	0.2167	0.1013	1.1129	0.1465	9.0079	0.1189	0.0628	0.0113	0.1310	0.3148	11.2940	0.0914	0.0466	0.0115	0.1217	0.0759	9.7898
DGCF	0.2791	0.2231	0.1047	1.1490	0.1462	**9.0111**	0.1264	0.0674	0.0123	0.1400	0.2483	10.8904	**0.1046**	0.0536	**0.0132**	**0.1407**	0.0599	9.6502
LR-GCCF	**0.2876**	**0.2274**	**0.1056**	**1.1589**	0.1245	8.7438	0.1246	0.0664	0.0119	0.1388	0.4037	11.6542	0.0990	0.0504	0.0124	0.1377	0.1367	10.8977
`Implicit` message-passing																		
UltraGCN	0.2540	0.2045	0.0901	0.9921	0.0766	8.0334	0.1256	0.0675	0.0123	0.1382	0.1737	10.0458	0.1041	**0.0541**	0.0131	0.1397	0.0586	9.0948
GFCF	0.1685	0.1398	0.0583	0.6577	0.0117	5.4064	**0.1287**	**0.0744**	**0.0137**	**0.1544**	0.2392	10.4923	0.0946	0.0496	0.0115	0.1158	0.0277	7.5926

accuracy performance on Amazon Digital Music and Epinions, respectively, with GFCF outperforming all other baselines on novelty, and UltraGCN getting slightly lower EPC and EFD values than DGCF). Unexpectedly, NGCF settles as the approach producing the most diverse lists of recommended items on all datasets (i.e., see Gini and SE) but cannot cope with the other baselines in terms of Recall and nDCG (similarly to Random). Other graph models with **`explicit`** message-passing (especially DGCF and LR-GCCF) are placed in the best accuracy/diversity trade-off spot, as they are often the second-to-best approaches on diversity, with limited observable drops in the accuracy. Contrarily, techniques with **`implicit`** message aggregation always show the lowest diversity.

Observation 1. *While the accuracy/novelty trade-off does not depend on the explicit/implicit message-passing, the accuracy/diversity trade-off is preserved only when explicitly propagating messages, at the expense of recommendation accuracy drops.*

A finer trade-offs evaluation. Figure 6.8 shows the accuracy/novelty/diversity trade-off on Amazon Digital Music by varying the message-passing strategy (i.e., **`explicit`** and **`implicit`**) and neighbor exploration depth only for the former case. Specifically, we use the reformulation from Section 6.4.2 to separate explicit message propagation results into **`same`**- and **`different`**-type node explorations at 1/2 hops.

We confirm that, while UltraGCN and GFCF can compete well on the accuracy/novelty trade-off with the other baselines (whatever the explored number of hops and node type), the opposite occurs on the accuracy/diversity trade-off. Indeed, higher accuracy values for UltraGCN and GFCF are obtained at the expense of significant drops in their diversity, even compared to message propagation at 1 hop (e.g., DGCF surpasses them on diversity at the expense of a slightly lower accuracy in the **`same`**-node).

As for the influence of **`same`**- and **`different`**-type node explorations, wider explorations of the neighborhood almost always lead to improved accuracy/novelty and

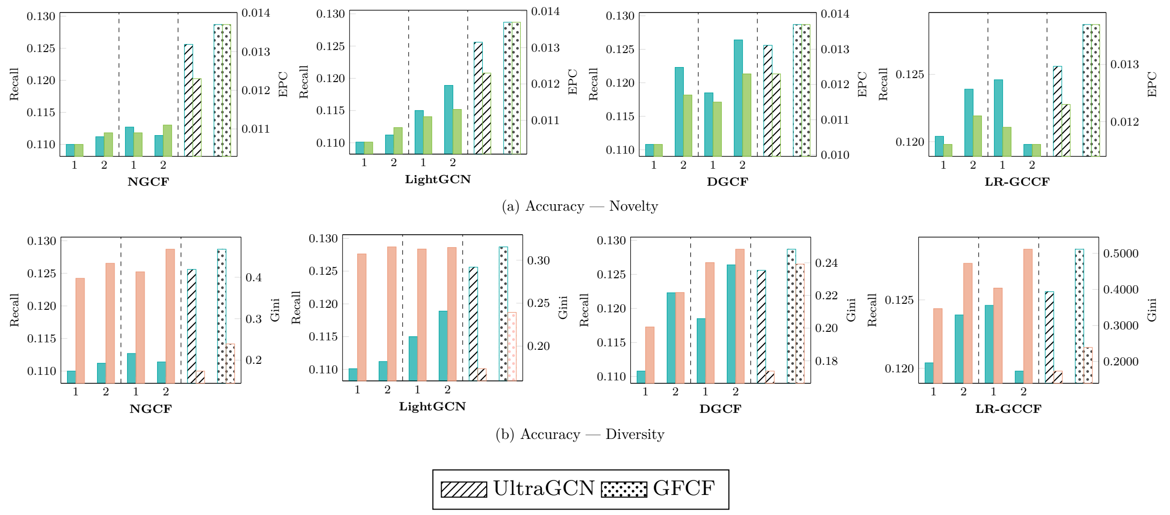


Fig. 6.8 Accuracy/Novelty (a) and Accuracy/Diversity (b) trade-offs of graph models with **`explicit`** (i.e., filled bar plots) and **`implicit`** message-passing (i.e., patterned bar plots) on Amazon Digital Music for top-20 recommendation lists. As for explicit message-passing, results are further categorized into **`different`**- and **`same`**-node type explorations (i.e., the leftmost and central tabs in each plot, respectively), when varying the number of hops from 1 to 2. Accuracy, novelty, and diversity are assessed through Recall (in teal blue), EPC (in lime green), and Gini (in melon), respectively. Best viewed in color.

accuracy/diversity performance, independently of the explored node types (apart from the **`same`**-type settings for NGCF on the Recall and LR-GCCF on the Recall and the EPC). Noticeably, the exploration of 1 hop in the **`same`**-type node setting leads to a better trade-off in accuracy/novelty/diversity than the exploration of 2 hops in the **`different`**-node setting (e.g., LightGCN increases the Recall and the EPC without a significant variation of Gini, and DGCF slightly decreases the Recall and the EPC, but improves Gini).

Observation 2. *To confirm observation 1, explicit message propagation (even at 1 hop) can reach a better accuracy/diversity trade-off than implicit propagation; same-type node explorations lead to improved accuracy/novelty and accuracy/diversity trade-offs.*

6.5 Auditing consumer- and provider-fairness

The adoption of deep learning (and, often, black-box) approaches to the recommendation task has raised issues regarding the fairness of recommender systems. The concept of fairness in recommendation is multifaceted. Specifically, the two core aspects to categorize recommendation fairness may be summarized as (1) the primary parties engaged (consumers vs. providers) and (2) the type of benefit provided (exposure vs.

relevance). Item suppliers are more concerned about exposure fairness than customers because they want to make their products better known and visible (**P**rovider fairness). However, from the customer's perspective, relevance fairness is of utmost importance, and hence system designers must ensure that exposure of items is equally effective across user groups (**C**onsumer fairness). A recent study highlights that papers on recommendation fairness concentrated on either C-fairness or P-fairness [224], disregarding the joint evaluation between C-fairness, P-fairness, and the accuracy.

The various graph CF *strategies* described above have historically centered on the enhancement of system accuracy, but, actually, never focused on the recommendation fairness dimensions. Despite some recent graph-based approaches have specifically been designed to address C-fairness [99, 172, 252, 315, 319, 345] and P-fairness [37, 213, 215, 292, 395, 401], there is a notable *knowledge gap* in the literature about the effects of the state-of-the-art graph *strategies* on the three objectives of C-fairness, P-fairness, and system accuracy. This study intends to complement the previous research and provide answers to pending research problems such as how different graph models perform for the three evaluation objectives. By measuring these dimensions in terms of **overall accuracy**, **user fairness**, and **item exposure**, we observe these aspects[17].

Motivating example. A preliminary comparison of the leading graph and classical CF models is carried out to provide context for our study. The graph-based models include LightGCN [126], DGCF [328], LR-GCCF [61], and GFCF [278], which are tested against two classical CF baselines, namely BPRMF [258] and RP$^3\beta$ [246], on the Baby, Boys & Girls, and Men datasets from the Amazon catalog [228]. We train each baseline using a total of 48 unique hyper-parameter settings and select the optimal configuration for each baseline as the one achieving the highest accuracy on the validation set (as in the original papers). Overall accuracy, user fairness, and item exposure (as introduced above) are evaluated. Figure 6.9 displays the performance of the selected baselines on the three considered recommendation objectives. For better visualization, all values are scaled between 0 and 1 using min-max normalization, and, when needed, they are replaced by their 1's complement to adhere to the "higher numbers are better" semantics. As a result, in each of the three dimensions, the values lay in $[0, 1]$ with higher values indicating the better. Please, note that such an experimental evaluation is not the main focus of this analysis but it is the motivating example for the more extensive analysis we present later. The interested reader may refer to Section 6.5.3

[17]In the rest, when no confusion arises, we will refer to C-fairness with user fairness, to P-fairness with item exposure, and to their combination as CP-fairness.

for a presentation of the full experimental settings to reproduce these results and the ones reported in the following sections.

First, according to Figure 6.9, graph CF models are significantly more accurate than the classical CF ones, even if the latter perform far better in terms of item exposure. Moreover, the displayed trends suggest there is no clear winner on the user fairness dimension: classical CF models show promising performance, while some graph CF models do not achieve remarkable results. As a final observation, an underlying trade-off between the three evaluation goals seems to exist, and it might be worth investigating it in-depth. Such outcomes open to a more complete study on how **different strategy patterns** recognized in graph CF may affect the three recommendation objectives, which is the scope of this study.

Research questions and contributions. In the remainder, we therefore attempt to answer the following two research questions (RQs):

RQ1. Given the different graph CF strategies, the raising question is *"Can we explain the variations observed when testing several graph models on overall accuracy, item exposure, and user fairness separately?"* According to a recent benchmark that identifies some state-of-the-art graph techniques [409], the suggested graph CF taxonomy (Table 6.17) extends the set of graph-based models introduced in the motivating example by examining eight state-of-the-art graph CF baselines through their strategies for *nodes representation* and *neighborhood exploration.* We present a more nuanced view of prior findings by analyzing the impact of each taxonomy dimension on overall accuracy and CP-fairness.

RQ2. The demonstrated performance prompts the questions: *"How and why nodes representation and neighborhood exploration algorithms can strike a trade-off between overall accuracy, item exposure, and user fairness?"* We employ the Pareto optimality to determine the influence of such dimensions in two-objective scenarios, considering overall accuracy, item exposure, and user fairness. The Pareto frontier is computed for three 2-dimensional spaces: accuracy/item exposure, accuracy/user fairness, and item exposure/user fairness.

6.5.1 A formal taxonomy of graph CF

Updating node representation through message-passing

The representation of users' and items' nodes are updated by leveraging the graph topology from $\mathcal{G}$. In this respect, the message-passing schema has recently gained

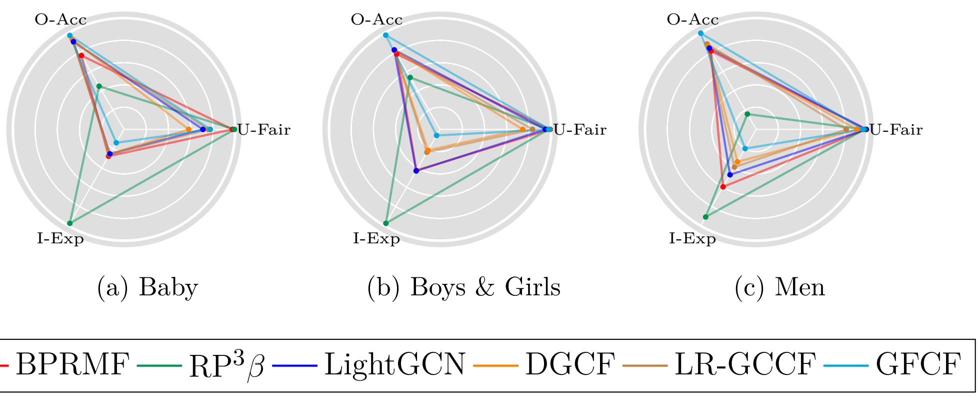


Fig. 6.9 Kiviat diagrams indicating the performance of selected pure and graph CF recommenders on overall accuracy (i.e., O-Acc, calculated with the nDCG@20), item exposure (i.e., I-Exp, calculated with the APLT@20 [3]), and user fairness (U-Fair, calculated with the UMADrat@20 [81]). Higher means better.

Table 6.17 Categorization of the chosen graph baselines according to the proposed taxonomy. For each model, we refer to the technical description reported in the original paper and try to match it with our taxonomy.

Models	**Nodes Representation**				**Neighborhood Exploration**			
	Latent representation		**Weighting**		**Explored nodes**		**Message passing**	
	low	*high*	*weighted*	*unweighted*	*same*	*different*	*implicit*	*explicit*
GCN-CF* [158]		✓		✓	✓			✓
GAT-CF* [313]		✓	✓		✓			✓
NGCF [325]	✓			✓		✓		✓
LightGCN [126]	✓			✓		✓		✓
DGCF [328]	✓		✓			✓		✓
LR-GCCF [61]	✓			✓	✓	✓		✓
UltraGCN [217]	✓				✓	✓	✓	
GFCF [278]						✓	✓	

*The postfix -CF indicates that we re-adapted the original implementations (tailored for the task of node classification) to the task of personalized recommendation.

attention in the literature. The algorithm works by aggregating the information (i.e., the *messages*) from the *neighbor* nodes into the *ego* node, and the process is recursively performed for multiple hops thus exploring wider neighborhood portions. In general, the message-passing for l hops is:

$$\mathbf{e}_u^{(l)} = \omega\left(\left\{\mathbf{e}_{i'}^{(l-1)}, \forall i' \in \mathcal{N}(u)\right\}\right), \tag{6.34}$$

where $\omega(\cdot)$ and $\mathcal{N}(\cdot)$ are the aggregation function and neighborhood node set, respectively, while l is in $1 \leq l \leq L$, where L is a hyper-parameter. Note that the following statements hold: $\mathbf{e}_u^{(0)} = \mathbf{e}_u$ and $\mathbf{e}_i^{(0)} = \mathbf{e}_i$. A reworking of Equation (6.34) for $l \in \{2,3\}$

allows *same-* and *different*-type node representation emerge [18]:

$$
\begin{array}{ll}
\textbf{\textit{Same-type node representation}} & \left\{ \underbrace{\mathbf{e}_u^{(2)}}_{\text{(user)}} = \omega\Big(\Big\{\omega\Big(\Big\{ \underbrace{\mathbf{e}_{u''}^{(0)}}_{\text{(user)}}, \forall u'' \in \mathcal{N}(i') \setminus \{u\}\Big\}\Big), \forall i' \in \mathcal{N}(u)\Big\}\Big) \right. \\
\textbf{\textit{Different-type node representation}} & \left\{ \begin{array}{l} \underbrace{\mathbf{e}_u^{(3)}}_{\text{(user)}} = \omega\Big(\Big\{\omega\Big(\Big\{\omega\Big(\Big\{ \underbrace{\mathbf{e}_{i'''}^{(0)}}_{\text{(item)}}, \forall i''' \in \mathcal{N}(u'') \setminus \{i''\}\Big\}\Big), \\ \qquad \forall u'' \in \mathcal{N}(i') \setminus \{u''\}\Big\}\Big), \forall i' \in \mathcal{N}(u)\Big\}\Big). \end{array} \right.
\end{array}
\tag{6.35}
$$

To better clarify the extent of Equation (6.35), after an **even** and an **odd** number of explored hops, *ego* node updates leverage by design *same-* and *different*-type node connections, i.e., user-user/item-item and user-item/item-user as evident from Equation (6.35). While the existing literature does not always consider the two scenarios as distinct, we underline the importance of investigating the influence of different node-node connections explored during the message-passing. In light of the above, we will count the number of explored hops as follows: $\mathbf{e}_*^{(2l)}, \forall l \in \{1, 2, \ldots, \frac{L}{2}\}$ as obtained through l **same-**type node connections (denoted as *same-l*), and $\mathbf{e}_*^{(2l-1)}, \forall l \in \{1, 2, \ldots, \frac{L}{2}\}$ as obtained through l **different-**type node connections (denoted as *different-l*). In the following, we introduce the graph convolutional network (GCN) and its CF applications.

The baseline: graph convolutional network (GCN). The standard graph convolutional network from Kipf et al. [158] performs feature transformation, message aggregation, application of a one-layer neural network, element-wise addition, and ReLU activation, respectively. Let us consider $\mathbf{W}^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ and $\mathbf{b}^{(l)} \in \mathbb{R}^{d_l}$ as the weight matrix and the bias for the l-th explored hop. Message-passing for user u is:

$$
\mathbf{e}_u^{(l)} = \mathrm{ReLU}\left(\sum_{i' \in \mathcal{N}(u)} \left(\mathbf{W}^{(l)} \mathbf{e}_{i'}^{(l-1)} + \mathbf{b}^{(l)}\right)\right). \tag{6.36}
$$

GCN for collaborative filtering. Inspired by the GCN message-passing approach, the authors from Wang et al. [325] propose neural graph collaborative filtering (NGCF). At each hop exploration, the model aggregates the neighborhood information and the inter-dependencies among the *ego* and the neighborhood nodes. Formally, the aggregation could be formulated as follows:

$$
\mathbf{e}_u^{(l)} = \mathrm{LeakyReLU}\left(\sum_{i' \in \mathcal{N}(u)} \left(\mathbf{W}_{\text{neigh}}^{(l)} \mathbf{e}_{i'}^{(l-1)} + \mathbf{W}_{\text{inter}}^{(l)} \left(\mathbf{e}_{i'}^{(l-1)} \odot \mathbf{e}_u^{(l-1)}\right) + \mathbf{b}^{(l)}\right)\right), \tag{6.37}
$$

where LeakyReLU is the activation function, $\mathbf{W}_{\text{neigh}}^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ and $\mathbf{W}_{\text{inter}}^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ are the neighborhood and inter-dependencies weight matrices, respectively, while $\odot$ is the Hadamard product.

He et al. [126] propose a light convolutional network, namely LightGCN, with the rationale to simplify the message-passing schema from GCN and NGCF by dropping feature transformations (i.e., the weight matrices and biases) and the non-linearity applied after the message aggregation. Specifically, they implement:

$$\mathbf{e}_u^{(l)} = \sum_{i' \in \mathcal{N}(u)} \mathbf{e}_{i'}^{(l-1)}. \tag{6.38}$$

The variation shows superior accuracy to the state-of-the-art. A slightly different solution [61] can outperform LightGCN regarding the accuracy level.

6.5.2 Weighting the importance of graph edges

The message-passing schema is inherently designed to aggregate into the *ego* node all messages coming from its neighborhood. Nevertheless, the *binary* nature of the user-item feedback (i.e., 0/1) would suggest that not all recorded user-item interactions necessarily hide the same importance to the nodes they involve.

In general, let $a_{y \to x}^{(l)}$ be the importance of the neighbor node y on its ego node x after l explored hops. We re-write the formulation of the message-passing after l explored hops (presented in Equation (6.34)) as:

$$\mathbf{e}_u^{(l)} = \omega\left(\left\{a_{i' \to u}^{(l)} \mathbf{e}_{i'}^{(l-1)}, \forall i' \in \mathcal{N}(u)\right\}\right). \tag{6.39}$$

The baseline: graph attention network (GAT). Attention mechanisms have reached considerable success in the GCN-related literature to weight the contribution of neighbor messages before aggregation. The original study [313] proposes the following message-passing formulation:

$$\begin{aligned} \mathbf{e}_u^{(l)} &= \sum_{i' \in \mathcal{N}(u)} \left(a_{i' \to u}^{(l)} \mathbf{W}_{\text{neigh}}^{(l)} \mathbf{e}_{i'}^{(l-1)} + \mathbf{b}^{(l)}\right) \\ &= \sum_{i' \in \mathcal{N}(u)} \left(\alpha\left(\mathbf{e}_{i'}^{(l-1)}, \mathbf{e}_u^{(l-1)}\right) \mathbf{W}_{\text{neigh}}^{(l)} \mathbf{e}_{i'}^{(l-1)} + \mathbf{b}^{(l)}\right), \end{aligned} \tag{6.40}$$

where $\alpha(\cdot)$ is the importance function depending on the lastly-calculated embeddings of the neighbor and the ego nodes, e.g., $a_{i' \to u}^{(l)} = \alpha\left(\mathbf{e}_{i'}^{(l-1)}, \mathbf{e}_u^{(l-1)}\right)$.

GAT for collaborative filtering. The authors from Wang et al. [328] design a message-passing schema that calculates the importance of neighborhood nodes for *ego* nodes by disentangling the intents underlying each user-item interaction. Similarly to He et al. [126] and Chen et al. [61], they therefore propose the following embedding update formulation:

$$\begin{aligned}\mathbf{e}_u^{(l)} &= \sum_{i' \in \mathcal{N}(u)} a_{i' \to u}^{(l)} \mathbf{e}_{i'}^{(l-1)} \\ &= \sum_{i' \in \mathcal{N}(u)} \alpha \left(\mathbf{e}_{i'}^{(l-1)}, \mathbf{e}_u^{(l-1)}, K, T\right) \mathbf{e}_{i'}^{(l-1)},\end{aligned} \tag{6.41}$$

where $\alpha(\cdot, K, T)$ is the importance function of the lastly-calculated embeddings from the neighbor and the *ego* nodes, e.g., $a_{i' \to u}^{(l)} = \alpha\left(\mathbf{e}_{i'}^{(l-1)}, \mathbf{e}_u^{(l-1)}, K, T\right)$, K is the total number of intents, and T is the total number of routing iterations to repeat the disentangling procedure.

Going beyond message-passing

The recent graph learning literature [54, 405] has outlined the phenomenon of *over-smoothing*, that leads node representations to become more similar as more hops are explored. The issue is generally tackled by limiting the neighborhood exploration to (maximum) three hops, and to two hops when attention mechanisms are introduced. However, the idea of improving accuracy by restricting the number of explored neighborhoods is counter-intuitive and "conflicts" with the rationale behind collaborative filtering [19]. This awareness led works such as Mao et al. [217] and Shen et al. [278] to surpass and simplify the traditional concept of message-passing. UltraGCN [217] adopts negative sampling to contrast over-smoothing and additional objective terms to (i) approximate the infinite neighborhood exploration and (ii) mine relevant "unexpected" node-node interactions such as the item-item ones. Conversely, GFCF [278] translates the graph-based recommendation task into the graph signal processing domain to obtain a closed-form formulation for approximating the infinite neighborhood exploration. Given that such recent strategies do not *explicitly* perform the message-passing schema as presented above, in the remaining sections, we will adopt the terms *explicit* and *implicit* message-passing as shorthands to denote the two model families, respectively.

A taxonomy of graph CF approaches

We propose (see Table 6.17) a taxonomy to classify the state-of-the-art graph models. The taxonomy considers the recurrent **strategy patterns** as emerged by conducting an in-depth review and analyzing the different graph CF approaches.

- **Node representation** indicates the representation strategy to model users' and items' nodes. It involves the *dimensionality* of node embeddings, and the possibility of *weighting* the neighbor node contributions.
- **Neighborhood exploration** refers to the procedure for exploring the multi-hop neighborhoods of each node to update the node latent representation. It involves the type of *node-node connections* which are explored, and the *message-passing* schema (i.e., *explicit* or *implicit* as previously defined).

In the next two sections, we will assess the performance of the graph CF models from the taxonomy in Table 6.17. Thus, we consider GCN-CF [158], GAT-CF [313], NGCF [325], LightGCN [126], DGCF [328], LR-GCCF [61], UltraGCN [217], and GFCF [278] for a total of eight graph CF solutions.

6.5.3 Experimental settings and protocols

Datasets. As a pre-processing stage, for each dataset, we randomly sample 60k interactions and drop users and items with less than five interactions to avoid the cold-start effect [124, 125]. The final dataset statistics are: (1) Baby has 5,842 users, 7,925 items, 35,475 interactions; (2) Boys & Girls has 3,042 users, 12,912 items, 35,762 interactions; (3) Men has 3,909 users, 27,656 items, 51,519 interactions.
Reproducibility. Datasets are split using the 70/10/20 train/validation/test hold-out strategy. Baselines are trained through grid search (48 explored configurations), with a batch size of 256 and 400 epochs. Datasets and codes (implemented with Elliot [11]) are available at: https://github.com/sisinflab/ECIR2023-Graph-CF.
Evaluation. As for the *overall accuracy*, we use the recall (Recall@k) and the normalized discounted cumulative gain (nDCG@k). Concerning the *item exposure*, we focus on: (1) item novelty [310, 311] through the expected free discovery (EFD@k) measuring the expected portion of relevantly-recommended items that have already been seen by the users; (2) item diversity [276] with the 1's complement of the Gini index (*Gini*@k), a statistical dispersion measure which estimates how a model suggests heterogeneous items to users; (3) the average percentage of items from the long-tail (APLT@k) which are recommended in users' lists [3] to calculate recommendation's bias towards popular items. *User fairness* indicates how equally each user group receives accurate recommendations. Users are split into quartiles based on the number of items they interacted with. We then measure UMADrat@k and the UMADrank@k [81], where the former stands for the average deviation in the predicted ratings among users groups, while the latter represents the average deviation in the recommendation accuracy

(calculated in terms of *nDCG@k*) among users groups. The best hyper-parameter configurations are found by considering Recall@20 on the validation.

6.5.4 Taxonomy-aware evaluation

This section aims to answer RQ1 (*"Can we explain the variations observed when testing several graph models on overall accuracy, item exposure, and user fairness separately?"*) by showing how the proposed taxonomy of graph strategies can explain the recommendation evaluation on CP-Fairness and overall accuracy. We experiment with 48 hyper-parameter configurations to investigate various combinations of graph CF techniques for *message-passing*, *explored nodes*, *edge weighting*, and *latent representations.* Results refer to the Amazon Men dataset and top-20 lists (Table 6.18). Please note that we report the **best** metric result for each <dimension, value> pair (the corresponding best graph recommendation model is displayed below each metric result) to ease the interpretation of results and provide meaningful insights.

- **Message-passing.** We investigate the two widely-recognized message-passing strategies: *implicit* and *explicit.* The most obvious pattern indicates that both sets have almost the same number of top-performing models in each of the evaluation criteria. *Explicit* graph approaches perform better on item exposure, where they outperform *implicit* techniques (i.e., on Gini and APLT) two out of three times by a significant margin. On the one hand, this tendency may be due to the absence of a direct message (information) propagating along the user-item graph in *implicit* techniques, which prevents the user node from exploring vast item segments. On the other hand, it appears that models from both families perform similarly on accuracy and user fairness, indicating that there is no obvious reason to favor *implicit* over *explicit* or vice versa.
- **Explored nodes.** Here, we examine four methods to explore nodes (adopting the message-passing re-formulation from Equation (6.35)): *same* and *different*, with 1 and 2 hops. Similarly to the trend found for the message-passing dimension, the results demonstrate that the two primary categories (*same* and *different*) are nearly equally performing across all measurements, with *same-2* and *different-1* being the prominent ones. In detail, the *different-1* exploration outperforms the *same-2* on the overall accuracy level (GFCF is the leading model here). Conversely, *same-2* is the best strategy for item exposure (with LR-GCCF and GAT-CF leading). As observed for the message-passing, user fairness does not give a reason to choose

Table 6.18 Best metric results (and corresponding graph CF model) for each <dimension, value> pair, on the Amazon Men dataset for top-20 lists. **Bold** is used to indicate the best result in the pairs having a two-valued dimension, while † is used only for the "explored nodes" dimension to indicate also the best results on *same* and *different*. The symbols $\uparrow$ and $\downarrow$ indicate whether better stands for high or low values. We use "rank" and "rat" as the UMADrank@k and UMADrat@k.

Dimensions	**Values**	**Overall Accuracy**		**Item Exposure**			**User Fairness**	
		Recall$\uparrow$	nDCG$\uparrow$	EFD$\uparrow$	Gini$\uparrow$	APLT$\uparrow$	rank$\downarrow$	rat$\downarrow$
Message passing	*implicit*	0.1222 (GFCF)	**0.0911 (GFCF)**	**0.2615 (GFCF)**	0.2871 (UltraGCN)	0.1808 (UltraGCN)	0.0123 (UltraGCN)	**0.0022 (UltraGCN)**
	explicit	**0.1223 (LR-GCCF)**	0.0884 (LR-GCCF)	0.2536 (LR-GCCF)	**0.5090 (LR-GCCF)**	**0.3823 (GAT-CF)**	**0.0002 (DGCF)**	0.0169 (LightGCN)
Explored nodes	*same-1*	0.1221† (LR-GCCF)	0.0884† (LR-GCCF)	0.2500† (LR-GCCF)	0.4377 (LR-GCCF)	0.3433 (GAT-CF)	**0.0002† (DGCF)**	**0.0022† (UltraGCN)**
	same-2	0.1184 (LightGCN)	0.0841 (LightGCN)	0.2380 (LightGCN)	**0.5090† (LR-GCCF)**	**0.3823† (GAT-CF)**	**0.0002† (DGCF)**	0.0209 (NGCF)
	different-1	**0.1222† (GFCF)**	**0.0911† (GFCF)**	**0.2615† (GFCF)**	0.4093 (NGCF)	0.3424 (GAT-CF)	**0.0002† (DGCF)**	**0.0022† (UltraGCN)**
	different-2	0.1210 (DGCF)	0.0850 (DGCF)	0.2407 (LightGCN)	0.4934† (LR-GCCF)	0.3438† (LR-GCCF)	**0.0002† (DGCF)**	0.0388 (LightGCN)
Weighting	*weighted*	0.1210 (DGCF)	0.0857 (DGCF)	0.2428 (DGCF)	0.3240 (DGCF)	**0.3823 (GAT-CF)**	**0.0002 (DGCF)**	0.0301 (DGCF)
	unweighted	**0.1223 (LR-GCCF)**	**0.0884 (LR-GCCF)**	**0.2536 (LR-GCCF)**	**0.5090 (LR-GCCF)**	0.3438 (LR-GCCF)	0.0101 (GCN-CF)	**0.0169 (LightGCN)**
Latent representations	*emb-64*	0.1193 (LR-GCCF)	0.0871 (LR-GCCF)	0.2479 (LR-GCCF)	**0.5090 (LR-GCCF)**	0.3627 (GAT-CF)	**0.0002 (DGCF)**	0.0054 (UltraGCN)
	emb-128	0.1221 (LR-GCCF)	0.0883 (LR-GCCF)	**0.2536 (LR-GCCF)**	**0.5090 (LR-GCCF)**	0.3644 (GAT-CF)	**0.0002 (DGCF)**	0.0111 (UltraGCN)
	emb-256	**0.1223 (LR-GCCF)**	**0.0884 (LR-GCCF)**	0.2532 (LR-GCCF)	0.5038 (LR-GCCF)	**0.3823 (GAT-CF)**	**0.0002 (DGCF)**	**0.0022 (UltraGCN)**

between *same* and *different*. The exploration of 1 hop in *same* and *different* settings is the preferable technique, even if 2 hops connections lead to a better item exposure.

- **Weighted.** This study examines *weighted* and *unweighted* graph CF techniques. Differently from above, we observe that *unweighted* solutions provide the best performance on almost all CP-fairness metrics, with LR-GCCF steadily being the superior approach. The only trend deviation refers to GAT-CF (i.e., a *weighted* method) surpassing *unweighted* solutions on the APLT level, that is, recommending items from the long-tail. The behavior is likely attributable to the design of *weighted* techniques, which can investigate farther neighbors of the *ego* node (observe the performance of GAT-CF on the *same-2* dimension), leading user profiles to match distant (and possibly niche) products in the catalog. On the contrary, it is interesting to notice how the other two metrics accounting for item exposure (i.e., EFD as item novelty measure and Gini as item diversity measure) seem to privilege *unweighted* graph techniques (i.e., LR-GCCF). The observed behaviors differ as the three metrics provide completely different perspectives of the *item exposure*, and thus they are uncorrelated.

- **Latent representations.** We compare the performance of graph CF techniques adopting latent representations with *64*, *128*, and *256* features, respectively. It is worth noticing that higher-dimensional latent representations (i.e., *128* and *256*) result in better performance on all measurements. Specifically, it appears that the *128* dimension is the turning point after which the trend becomes stable (i.e., the metric values for *128* and *256* are frequently comparable). This may be an important insight since the majority of research works in recent literature tend to employ *64*-embedded representations of nodes without exploring further dimensionalities (see Table 6.17 as a reference).

6.5.5 Trade-off analysis

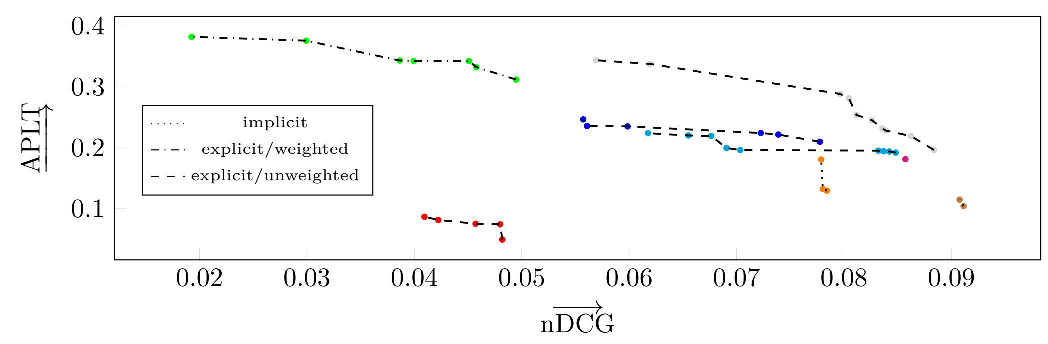
This section analyses how the graph CF baselines balance the trade-off among accuracy, item exposure, and user fairness, and aims to answer RQ2 (*"How and why nodes representation and neighborhood exploration algorithms can strike a trade-off between overall accuracy, item exposure, and user fairness?"*). Due to space constraints, we report the results only for the Amazon Men dataset. The negative Pearson correlation values for accuracy/item exposure (nDCG/APLT) and accuracy/user fairness (nDCG/UMADrank) suggest that a trade-off may be necessary, and desirable. In addition, the same correlation metric indicates the necessity of a trade-off for item exposure/user fairness (APLT/UMADrank). Among the strategy patterns identified in the proposed taxonomy (see Table 6.17), we select the most important architectural dimensions, **message-passing** and **weighting** of graph edges, to conduct this study. In detail, the analysis studies three combined categories: (1) models with implicit message-passing (denoted as *implicit*); (2) models with explicit message-passing and neighborhood weighting (denoted as *explicit/weighted*); (3) models with explicit message-passing without neighborhood weighting (denoted as *explicit/unweighted*). For each analyzed trade-off, we select the Pareto optimal solutions[18] of the baselines laying on the model-specific Pareto frontier [243]. Figure 6.10 plots graph models Pareto frontiers in the common *objective function spaces* related to the considered trade-offs. The careful

[18]A solution is Pareto optimal if no other solution can improve an objective without hurting the other one.

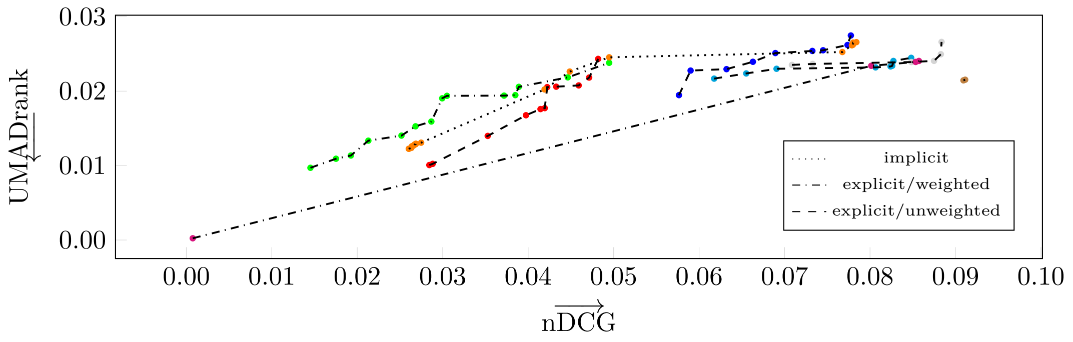
reader may notice the different axis' scales across the graphics due to the metric values. The colors of Pareto optimal solutions are model-specific, while the line style is used to distinguish the categories: dotted lines for *implicit*, dash-dot lines for *explicit/weighted*, and dashed lines for *explicit/unweighted.*

- **Accuracy/Item Exposure.** Figure 6.10a shows that the *explicit/weighted* models exhibit a trade-off, as they maximize either nDCG (i.e., DGCF) or APLT (i.e., GAT-CF), but not both. This is expected since DGCF is designed as a version of GAT-CF with improved accuracy. It is worth mentioning that DGCF's trade-off is reached at the expense of item exposure. In contrast to these models, *explicit/unweighted* baselines show a balanced trade-off because they do not prioritize accuracy or item exposure exclusively. In detail, LR-GCCF provides the best performance in terms of nDCG and APLT simultaneously. From a visual inspection, LR-GCCF's Pareto frontier dominates those of the other *explicit/unweighted* models. Conversely, GCN-CF exhibits the worst trade-off because it is neither ideal for nDCG nor APLT. As for the *implicit* models, they appear to prioritize precision over the provision of long-tail items. *Under this lens, the latest (i.e., implicit) approaches seem to increase accuracy, even if this is to the detriment of the niche items exposure.*

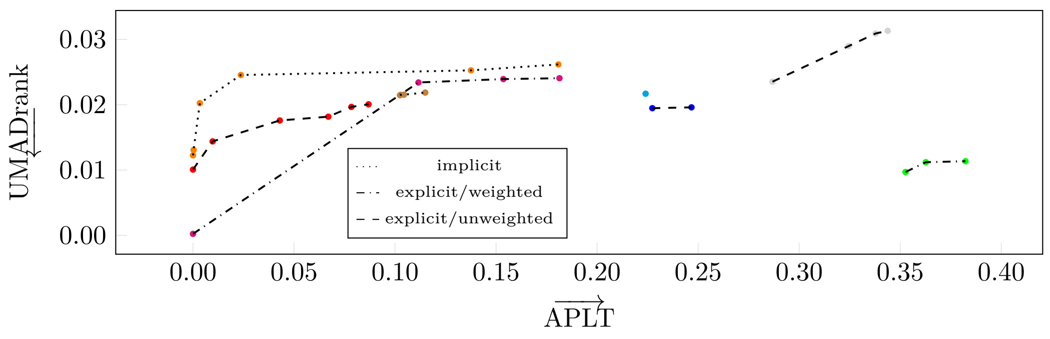
- **Accuracy/User Fairness.** To ease the interpretation of Figure 6.10b, we recall that UMADrank (used to measure User Fairness) measures to what extent the model ranking performance differs among the user groups (partitioned based on their activity on the platform). Figure 6.10b shows that, for GAT-CF and GCN-CF, the poor performance in terms of nDCG is associated with high variability in terms of user fairness. In fact, for these two models, the UMADrank value indicates high variability across user groups. Something different emerges for models such as NGCF, LightGCN, LR-GCF, and GFCF. These models, GFCF in particular, exhibit valuable recommendation accuracy with better stability in terms of ranking performance across the different user groups. As a consequence, the Pareto frontiers associated with these models dominate the others. In detail, GFCF is the best-performing one regarding this trade-off. Conversely, UltraGCN and DCGF do not show consistent behavior demonstrating a strong sensitivity to the chosen hyper-parameters set. *In this setting, no graph CF strategy emerges as the absolute winner. Specifically, every graph CF strategy is not enough to guarantee adequate fairness among different user groups. Then, the positive results are associated with particular configurations of some models and are lost when the hyper-parameter set changes.*



(a) Overall Accuracy/Item Exposure



(b) Overall Accuracy/User Fairness



(c) Item Exposure/User Fairness

● GCN-CF ● GAT-CF ● NGCF ● LightGCN
● DGCF ● LR-GCCF ● UltraGCN ● GFCF

Fig. 6.10 Overall Accuracy/Item Exposure, Overall Accuracy/User Fairness, and Item Exposure/User Fairness trade-offs on Amazon Men, assessed through nDCG/APLT, nDCG/UMADrank, and APLT/UMADrank, respectively. Each point depicts a model hyper-parameter configuration set belonging to the corresponding Pareto frontier. Colors refer to a particular baseline, while lines styles discern their technical strategies based on the proposed taxonomy. Arrows indicates the optimization direction for each metric on x and y axes.

- **Item Exposure/User Fairness.** The trade-off indicates to what extent graph CF models can treat final users fairly and recommend items from the long tail. In Figure 6.10c, it is possible to identify two groups of baselines: the models that show poor performance in terms of item exposure (UltraGCN, DGCF, GCN-CF, and GFCF) and the models that exhibit an acceptable exposure for long-tail items (LightGCN, NGCF, LR-GCCF, and GAT-CF). In detail, a cluster of models that belong to the *explicit/unweighted* category stands out in this second group. Not only are these models able to recommend niche items, but also they are stable (among the user groups) in terms of accuracy. On the contrary, although GAT-CF lies close to the *utopia point*[19], it exhibits greater variability regarding the accuracy metric. Indeed, comparing Figure 6.10c with Figure 6.10a, GAT-CF demonstrates to achieve adequate user fairness, but its performance is still very poor in terms of accuracy. *To summarize, even if a system designer could be more interested in promoting models solely guaranteeing the best value for APLT (provider Fairness), the explicit/unweighted strategies can generally ensure a satisfactory (for Consumers and providers) trade-off between user fairness and item exposure.*

6.6 Summary

This chapter tailored the experimental and evaluation paradigms introduced in the previous chapters to the state-of-the-art strategies and techniques for graph-based recommendation, also defined under the paradigm of graph collaborative filtering. Similarly to the previous chapter, we proposed a framework for the reproducibility and evaluation of graph-based recommender systems built upon Elliot, later used to replicate the results of such models. Indeed, the reproducibility analysis uncovered interesting and unexpected findings regarding the possible influence of dataset characteristics on the performance of graph-based recommender systems. Thus, we decided to extend the investigation by considering some of the main topological properties of the user-item graph in terms of clustering coefficient and degree assortativity (conceptually linked to node degree). Results showed how such dependences are strongly evident in graph-based recommendation, and their latent-factor component may be the one component influencing recommendation performance the most. Indeed, this helped re-interpreting the graph collaborative filtering strategies under another perspective which might drive the designing and implementation of novel such approaches according to graph topological characteristics of each dataset. Finally, we studied how and to what extent

[19]The point that simultaneously minimizes (maximizes) all the metrics.

existing strategies adopted for each step of the recommendation pipeline in graph-based recommendation may influence the performance in terms of novelty, diversity, and consumer/provider fairness. Moreover, the proposed analysis was conducted in a single- and multi-objective settings. Observed results outlined that user-user and item-item message-passing explorations may improve accuracy/diversity/novelty trade-off; additionally, the outcomes on the consumer/provider fairness raised concerns about the effective application of recent approaches in graph collaborative (e.g., implicit message-passing techniques).

The current chapter helped providing the necessary background notions in terms of state-of-the-art, models' formulations, and (novel) performance evaluations regarding graph-based recommendation systems. Under such bases, and supported by the findings in the multimedia recommendation scenario previously considered, we present the final outcomes of this thesis. Thus, in the next chapter, we bring together the lessons-learned in multimedia recommendation and graph-based recommendation by studying and proposing novel approaches in graph-based recommender systems leveraging multimodal information.

Chapter 7

Graph-based recommendation exploiting multimodal information

This chapter seeks to combine the lessons-learned from multimodal-aware and graph-based recommendation with the proposal of a novel approach. Following the same experimental and evaluation paradigms already adopted in the previous chapters of the thesis, we first benchmark state-of-the-art (graph-based) models leveraging multimodal content on different recommendation dimensions with respect to the literature, namely, novelty, diversity, and popularity bias, along with various multimodality settings. Finally, by discussing the existing limitations in the way multimodality is generally handled in graph-based recommendation approaches, a new solution leveraging reviews on edges for graph-based recommendation is proposed and tested against state-of-the-art recommendation techniques leveraging reviews. Results demonstrate the efficacy of the introduced solution, which seems to be also able to limit the negative effects of known issues in graph learning, namely, the over-smoothing effect.

7.1 Novelty and diversity in multimodal-aware recommendation

Despite the indisputable success of multimodal-aware recommender systems as recognized in the recent literature, mainly thanks to their recommendation accuracy improvements to the existing baselines, some performance evaluation concerns still raise. Indeed, as most of these methods follow similar strategy patterns with few variations on the main theme, it can be challenging to unveil which technique is actually providing the most impactful contribution to the recommendation performance.

Additionally, most of such approaches are trained and evaluated under different implementations, which come with their own data splitting/sampling, hyper-parameter searching, and evaluation protocols.

To address such an evaluation gap in the literature, the first part of this chapter aims to provide the first extensive benchmarking setting for the multimodal-aware recommendation. Specifically, our contributions are threefold:

- We provide a unified framework to benchmark five state-of-the-art multimodal-aware recommender systems (i.e., VBPR [125], MMGCN [339], MGAT [305], GRCN [338], and LATTICE [382]) on three popular recommendation datasets from the Amazon catalog [218] (i.e., Office, Toys, and Clothing).
- We run extensive hyper-parameter explorations to fine-tune all tested models under the same settings for a fair comparison. While confirming some findings from the existing literature, results also show how careful hyper-parameter tuning can make even shallow approaches (e.g., VBPR) quite competitive against more complex and recent ones (e.g., GRCN).
- In addition to assessing recommendation accuracy, we complement the evaluation through an analysis of the novelty [310, 311] and diversity [276] of the produced lists of recommendation. To the best of our knowledge, this is the first attempt to perform this analysis in the domain of multimodal-aware recommendation.

To foster the reproducibility of our benchmark, we share the code and datasets adopted in this work: https://github.com/sisinflab/MultiModal-Eval.

7.1.1 Novelty and diversity in recommendation

User experience plays a crucial role in recommendation platforms, as highlighted by several academic studies [141, 149, 282]. Such works emphasize that suggesting interesting lists of items satisfies users and encourages them to remain loyal to the platform, ultimately leading to increased profits [316]. To ensure a good user experience, the recommended items must be nontrivial, diverse, possibly unexpected [110, 282], fair [75, 77], and explainable [76, 77]. However, designing dedicated models for recommendation systems presents significant challenges, mainly because evaluating them requires conducting user studies.

Consequently, researchers have invested substantial effort in exploring beyond-accuracy dimensions within the field of recommendation systems over the past two decades [276, 311, 379]. These dimensions refer to aspects beyond the traditional accuracy metric, aiming to improve the overall user experience.

While user experience has been a crucial aspect when evaluating multimodal-aware intelligent systems [147, 250, 390] for years, in recommendation it has gained attention only recently [35, 71]. As for multimodal-aware recommendation, most of the research efforts have focused on emphasizing the advantages of multimodal recommendation models in addressing the cold start user problem [125, 236, 242]. However, to the best of our knowledge, there is a lack of recent scientific literature that explicitly considers the impact of multimodality on user experience in terms of novelty and diversity of the produced recommendations. To this end, our analysis stands first and foremost as an attempt to bridge such an evaluation gap.

7.1.2 Proposed analysis

This section describes our proposed analysis for multimodal-aware recommendation. First, we report on the adopted datasets, along with details about the extraction of multimodal features. Then, we introduce and formalize the set of evaluation metrics accounting for accuracy, novelty, and diversity of recommendation. Finally, we outline the details about reproducibility for our work, by providing information about dataset splitting and filtering strategies, and the hyper-parameter search.

Datasets

In our study, we conduct extended experiments on three popular review datasets from the Amazon catalog [218] to better generalize the insight derived from our analysis. The categories are: Office Products (**Office**), (b) Toys & Games (**Toys**), and (c) Clothing, Shoes & Jewelry (**Clothing**). Each dataset consists of both visual and textual modalities, where the former are made available by McAuley et al. [218]. Thus, in our analysis, we utilize the already pre-extracted 4,096-dimensional visual features which are publicly available[1]. For the textual modality, by following [382], we aggregate the item's title, descriptions, categories, and brand, thereby generating textual embeddings. In this regard, we leverage sentence transformers [257, 382], acquiring 1,024-dimensional sentence embeddings. Additional dataset information can be found in Table 7.7.

[1]https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html.

Table 7.1 Statistics of the tested datasets.

Datasets	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{R}\rvert$	**Sparsity (%)**
Office	4,905	2,420	53,258	99.5513
Toys	19,412	11,924	167,597	99.9276
Clothing	39,387	23,033	278,677	99.9693

Evaluation metrics

Differently from the existing literature on multimodal-aware recommendation, we take into account metrics measuring the accuracy (Recall, nDCG, Precision), along with the novelty (EFD) and diversity (Gini and iCov) of recommendation.

Reproducibility

To ensure reproducibility, we provide detailed information about the dataset preprocessing and splitting, models' tuning and evaluation.

The datasets have been filtered following the p-core strategy with $p = 5$. Then, we split the dataset with 80%/20% train-test hold-out strategy. For the hyper-parameter tuning phase, we remove the 50% of the test set and use it to validate the results on Recall@20. For all models, we fix the maximum number of epochs to 200 and select the model weights according to the epoch providing the best results on the validation set.

The complete set of hyper-parameters is reported in Table 7.2. The code for the entire pipeline (which is implemented in Elliot [11]) can be found at this link https://github.com/sisinflab/MultiModal-Eval.

7.1.3 Results and discussion

In this section, we seek to answer the following research questions (i.e., RQs):

RQ1. *What is the accuracy performance of multimodal-aware recommender systems and is it aligned with the findings from the existing literature?* We investigate the recommendation performance in terms of *accuracy* (i.e., Recall, nDCG, and Precision).

RQ2. *What is the recommendation performance of such models in terms of novelty and diversity of the produced lists of recommendation?* We unveil important insights in terms of novelty and diversity of recommendation (i.e., iCov, Gini, and EFD).

Table 7.2 Set of explored and fixed hyper-parameters for our study.

Models	**Hyper-parameters**	**Values**
All	*epochs*	200
	batch_size	1024
	seed	123
VBPR	*lr*	{1e-2, 1e-3, 1e-4, 3e-5, 1e-5}
	factors	64
	comb_mod	concat
	l_w	{1e-5, 1e-2}
MMGCN	*lr*	{1e-2, 1e-3, 1e-4, 3e-5, 1e-5}
	num_layers	3
	factors	64
	factors_multimod	(256, None)
	aggregation	mean
	concatenation	False
	has_id	True
	latent_factors	64
	l_w	{1e-2, 1e-5}
MGAT	*lr*	{1e-2, 1e-3, 1e-4, 3e-5, 1e-5}
	num_layers	2
	factors	64
	factors_multimod	(256, 100)
GRCN	*lr*	{1e-2, 1e-3, 1e-4, 3e-5, 1e-5}
	num_layers	2
	num_routings	3
	factors	64
	factors_multimod	128
	aggregation	add
	weight_mode	confid
	fusion_mode	concat
LATTICE	*lr*	{1e-2, 5e-3, 1e-3, 5e-4, 1e-4}
	n_layers	1
	n_ui_layers	2
	top_k	20
	l_m	0.7
	factors_multim	64

Accuracy performance

The results of the *accuracy* metrics analysis is reported in Table 7.3. As a general remark, we notice how the results are quite standard across the different datasets.

Overall, LATTICE is the best model, showing its superior performance across all the datasets and the metrics, while VBPR is the second-to-best model. Surprisingly, complex and recent approaches such as MMGCN, MGAT, and GRCN do not outperform a shallow and classic model such as VBPR. Conversely, LATTICE's results are aligned with the findings from the literature.

From a *dataset-wise* analysis, the highest metrics-performance variation between LATTICE and VBPR is observable for Toys and Clothing while it is limited in Office. Indeed, we should point out that Toys and Clothing have three times and four times the interactions of Office, respectively, but they are much sparser. This highlights

Table 7.3 Accuracy results of the tested baselines when considering the top-10, top-20, and top-50 recommendation lists. **Boldface** and underline stand for best and second-to-best results on each dataset/metric pair, respectively.

Datasets	Models	$k = 10$			$k = 20$			$k = 50$		
		Recall	**nDCG**	**Prec**	**Recall**	**nDCG**	**Prec**	**Recall**	**nDCG**	**Prec**
Office	VBPR	0.0652	0.0419	0.0164	0.1025	0.0533	0.0133	0.1774	0.0721	0.0095
	MMGCN	0.0455	0.0300	0.0124	0.0798	0.0405	0.0109	0.1575	0.0598	0.0084
	MGAT	0.0427	0.0277	0.0119	0.0745	0.0377	0.0102	0.1450	0.0552	0.0079
	GRCN	0.0393	0.0253	0.0118	0.0667	0.0339	0.0099	0.1250	0.0488	0.0075
	LATTICE	**0.0664**	**0.0449**	**0.0173**	**0.1029**	**0.0566**	**0.0137**	**0.1780**	**0.0751**	**0.0096**
Toys	VBPR	0.0710	0.0458	0.0131	0.1006	0.0545	0.0096	0.1523	0.0667	0.0061
	MMGCN	0.0256	0.0150	0.0052	0.0426	0.0200	0.0044	0.0785	0.0285	0.0033
	MGAT	0.0375	0.0230	0.0072	0.0595	0.0294	0.0059	0.1039	0.0398	0.0043
	GRCN	0.0554	0.0354	0.0108	0.0831	0.0436	0.0083	0.1355	0.0559	0.0056
	LATTICE	**0.0805**	**0.0512**	**0.0148**	**0.1165**	**0.0617**	**0.0110**	**0.1771**	**0.0759**	**0.0069**
Clothing	VBPR	0.0339	0.0181	0.0034	0.0529	0.0229	0.0027	0.0847	0.0292	0.0017
	MMGCN	0.0227	0.0119	0.0023	0.0348	0.0150	0.0018	0.0609	0.0201	0.0012
	MGAT	0.0244	0.0127	0.0025	0.0384	0.0162	0.0019	0.0699	0.0225	0.0014
	GRCN	0.0319	0.0164	0.0032	0.0496	0.0209	0.0025	0.0858	0.0281	0.0017
	LATTICE	**0.0502**	**0.0275**	**0.0051**	**0.0744**	**0.0336**	**0.0038**	**0.1186**	**0.0425**	**0.0024**

how LATTICE manages to recommend more accurate items despite the high dataset sparsity.

From a *metric-wise* analysis, LATTICE, compared to VBPR, correctly predicts relevant items (i.e., high Recall) with a higher probability to be in a top positions of the recommendation lists (i.e., nDCG). The same trend is not observable in the Recall/Precision metric pair, but this is explainable considering that the latter formulation is normalized as the number of recommended items increases. Thus, it can result in a lower performance variation between LATTICE and VBPR at the increase of k.

From a *model-wise* analysis, we notice how MMGCN has better performance on Toys while showing the lowest performance at the increase of interactions number and sparsity. GRCN has an opposite trend compared to MMGCN, boosting its performance with highly sparse data. MGAT performs in the middle of MMGCN and GRCN with no remarkable note.

Summary. *Accuracy results demonstrate that, with the only exception of LATTICE (whose trend is almost aligned with the existing literature) all other approaches' performance is heavily influenced by the hyper-parameter exploration and dataset characteristics. Indeed, even shallow models (e.g., VBPR) show competitive accuracy measures compared to more recent and complex solutions (e.g., MMGCN, GRCN).*

Novelty/diversity performance

Table 7.4 summarizes the results of the *novelty* and *diversity* metrics analysis. Overall, we observe that some trends are quite aligned with findings from the accuracy evaluation, but also that some other ones show deviations which we carefully describe and explain.

On the one hand, in terms of recommendation *novelty* (i.e., EFD), we can see that LATTICE is the best model, with VBPR being the second-to-best approach in each dataset and for different settings of k. Indeed, this is a further demonstration on how accuracy and novelty of recommendation may be highly correlated [14, 17, 18], also in multimodal-aware recommendation.

On the other hand, when considering the *diversity* (i.e., Gini) and *coverage* (i.e., iCov) metrics, we notice some trends deviation to the accuracy performance. Specifically, we see how GRCN is the best model in all settings. This suggests that this approach may be (un)purposely giving up on the accuracy to promote a wider set of items from the catalog, with a corresponding positive effect on the system serendipity. Indeed, while its accuracy performance is not the best one, its diversity and coverage metrics outperform all other models almost on every dataset, even reaching 100% of covered items at $k = 50$. A much more impressive trend is recognizable for Gini, which is higher than the second-to-best model. On a dataset level, it is worth pointing out that, even with more sparse datasets, the GRCN constantly reaches a high iCov and Gini measures. The second-to-best model in terms of diversity is VBPR. Notwithstanding its high accuracy, VBPR settles once again as a compelling model.
Summary. *While novelty results are almost aligned with the accuracy trends observed in RQ1, the diversity/coverage measures depict a different scenario. In this respect, GRCN seems to be the approach providing the most diversified item recommendations but at the expense of the accuracy, while VBPR manages to reach a more balanced performance among all metrics.*

7.2 Multimodality and items' popularity bias

The vast majority of multimodal-aware recommender systems (MRSs) are generally based upon the famous matrix factorization with bayesian personalized ranking (MF-BPR) recommendation model. On the one hand, matrix factorization [160] (MF) is a latent-factor approach that maps users and items in the recommendation system to embeddings in the latent space and is trained to reconstruct the user-item interaction matrix via the dot product of the respective factors. On the other hand, bayesian personalized ranking [258] (BPR) is an optimization schema that drives from the

Table 7.4 Novelty and diversity results of the tested baselines when considering the top-10, top-20, and top-50 recommendation lists. **Boldface** and underline stand for best and second-to-best results on each dataset/metric pair, respectively.

Datasets	**Models**	$k = 10$			$k = 20$			$k = 50$		
		EFD	**Gini**	**iCov** (%)	**EFD**	**Gini**	**iCov** (%)	**EFD**	**Gini**	**iCov** (%)
Office	VBPR	0.1753	0.3634	93.83	0.1479	0.396	10.23	0.1115	0.4413	99.59
	MMGCN	0.1140	0.0128	3.07	0.1027	0.0231	4.64	0.0845	0.0546	10.23
	MGAT	0.1079	0.0132	5.14	0.0963	0.0241	8.12	0.0792	0.0575	17.23
	GRCN	0.1215	**0.4587**	**99.01**	0.1051	**0.4892**	**99.79**	0.0829	**0.5286**	**100**
	LATTICE	**0.1827**	0.2128	87.86	**0.1513**	0.2652	95.90	**0.1125**	0.3414	99.30
Toys	VBPR	0.1948	0.2645	84.90	0.1527	0.3011	92.82	0.1051	0.3585	97.85
	MMGCN	0.0648	0.0989	37.87	0.0570	0.1450	52.51	0.0455	0.2296	72.88
	MGAT	0.0929	0.1036	40.95	0.0796	0.1439	55.71	0.0612	0.2183	76.24
	GRCN	0.1604	**0.3954**	**92.66**	0.1298	**0.4329**	**97.73**	0.0932	**0.4864**	**99.73**
	LATTICE	**0.2090**	0.1656	73.80	**0.1665**	0.2026	86.58	**0.1151**	0.2662	95.94
Clothing	VBPR	0.0502	0.2437	83.40	0.0413	0.2791	92.33	0.0291	0.3344	98.00
	MMGCN	0.0292	0.0136	7.58	0.0240	0.0236	12.44	0.0182	0.0493	23.34
	MGAT	0.0315	0.0201	11.05	0.0263	0.0326	17.36	0.0205	0.0622	30.90
	GRCN	0.0481	**0.3990**	**93.37**	0.0397	**0.4368**	**97.77**	0.0293	**0.4929**	**99.73**
	LATTICE	**0.0738**	0.1022	58.49	**0.0589**	0.1384	76.20	**0.0413**	0.2037	93.23

assumption that, for each user, the predicted score of positive (i.e., interacted) and negative (i.e., non-interacted) items should diverge. Given its simple implementation and efficacy, MFBPR has long constituted the backbone of recommendation algorithms in CF [126, 128, 216], not only for multimodal recommendation.

Nevertheless, recommender systems (such as MFBPR) may be affected by popularity bias [3, 26, 39, 141] (Figure 7.1), as they tend to boost the recommendation of the items from the *short-head* (i.e., the popular ones) at the expense of the items from the *long-tail* (i.e., the niche ones). Tackling popularity bias in recommendation has primarily followed four directions [58]: (i) regularization techniques [3, 68, 150], (ii) adversarial learning [163], (iii) causal graphs [322, 387, 402], and (iv) other item re-ranking approaches [1, 4].

Despite the growing interest in popularity bias [17, 75] and potential solutions to address it, to date, very limited effort has been put into investigating ***how multimodal side information in MRSs could amplify the negative effects of popularity bias***. To the best of our knowledge, three recent works discussed the concept of bias in multimodal-aware recommendation. First, Liu et al. [192] take into account the bias towards a single modality in multimodal recommendation, and propose a solution based upon causal inference and counterfactual reasoning; however, the definition they provide about bias is conceptually different from the one of popularity bias. Then, Kowald et al. [162] consider popularity bias in the case of multimedia recommendation datasets (e.g., MovieLens); however, they do not support their findings by testing recommender

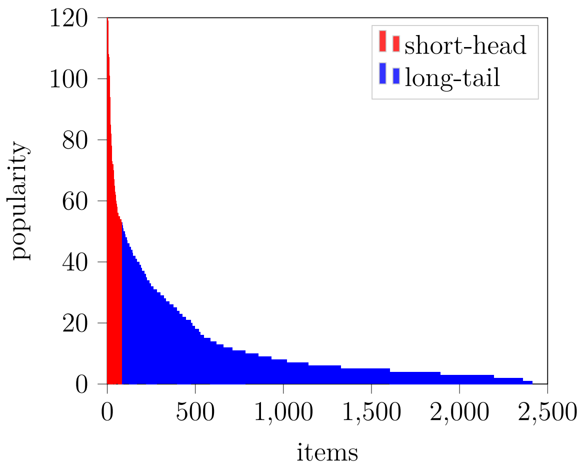


Fig. 7.1 Short-head and long-tail items from the *Office* dataset in the Amazon catalog.

systems leveraging multimodal features as items' side information. Last, Malitesta et al. [206] investigate how novelty and diversity metrics are influenced in multimodal recommendation, but without an analysis on the impact of each single modality.

Driven from the assumptions above, and differently from the related literature, we propose one of the first analyses on how multimodal-aware recommender systems may amplify popularity bias in the produced recommendation lists. To this aim, we select four established and recent multimodal-aware recommender systems from the literature (i.e., VBPR [125], MMGCN [339], GRCN [338], and LATTICE [382]) and train them on three categories of the Amazon recommendation dataset [218] (i.e., *Office*, *Toys*, and *Clothing*). Then, we evaluate the performance of the models by assessing metrics accounting for recommendation accuracy and popularity bias (the latter is measured through the diversity of recommendation lists and the percentage of retrieved items from the long-tail). Finally, to tailor our investigation, we focus on the separate impact of each multimodal side information (i.e., visual or textual) on popularity bias. To conduct this further study, we train the selected recommender systems when integrating either the visual or the textual modality as items' side information, and study the performance on single metrics and across pairs of metrics.

We seek to answer: **RQ1.** How do multimodal-aware recommendation models behave in terms of accuracy, diversity, and popularity bias? **RQ2.** What is the influence of each modality (i.e., visual, textual, multimodal) on such performance measures? Results widely show that the integration of a single modality (with respect to the multimodal setting) is capable of amplifying the negative effects of popularity bias, paving the way to additional, more formal investigations on multimodal recommendation. We release the code at: https://github.com/sisinflab/MultiMod-Popularity-Bias.

7.2.1 Popularity bias in recommendation

In recommendation, popularity bias refers to the system's tendency to favor popular items (i.e., *short-head*) at the expense of less popular ones (i.e., *long-tail*) [3, 26, 39, 43, 141]. For instance, Jannach et al. [141] conduct a comprehensive algorithmic comparison across multiple datasets; their findings indicate that existing recommendation methods tend to concentrate mainly on a small fraction of the available item spectrum. More recently, Abdollahpouri et al. [4] delve into this issue using the well-known MovieLens 1M dataset and reveal that over 80% of all ratings are attributed to popular items; their main focus lies in finding ways to strike a balance between ranking accuracy and the coverage of long-tail items.

On such basis, the literature currently recognizes four main research directions [58] to address popularity bias in recommendation, namely: (i) regularization techniques [3, 68, 150], (ii) adversarial learning [163], (iii) causal graphs [322, 387, 402], and (iv) other approaches such as item re-ranking [1, 4].

In multimodal recommendation, only a few recent works discuss popularity bias, but with specific definitions [192] and neglecting the impact of multimodal features [162], or on other evaluation metrics [206]. Conversely, our analysis assesses how prone multimodal-aware recommender systems are to push items belonging to the short-head and how the different modalities affect the tendency to amplify the popularity bias.

7.2.2 Factorization models leveraging multimodal information

This section provides useful background notions for our proposed experimental analysis. We present the formulations of four state-of-the-art multimodal-aware recommender systems (MRSs): VBPR [125], MMGCN [339], GRCN [338], and LATTICE [382]. Before diving into their approaches, we introduce some additional formalism.

Besides $\mathbf{e}_u$ and $\mathbf{e}_i$, hereafter referred to as *collaborative* user and item embeddings, we also introduce $\mathbf{f}_u$ and $\mathbf{f}_i$ as the *multimodal* embeddings for user u and item i. Moreover, we indicate $\mathcal{M}$ as the set of available modalities (e.g., visual, textual, audio), and we use m as embedding's apex to denote that the embedding refers to the $m \in \mathcal{M}$ modality (e.g., $\mathbf{f}_i^m$ stands for the m-th multimodal embedding of item i).

VBPR

Visual-bayesian personalized ranking [125] (dubbed as VBPR) adopts visual features extracted from product images as items' side information in MFBPR. The authors introduce, along with user and item *collaborative* embeddings, additional *visual* user and item embeddings, where the latter is obtained as the activation of the penultimate layer from a pre-trained convolutional neural network. Then, the collaborative and visual embeddings are used to measure a collaborative- and visual-aware prediction for the interaction score and are eventually summed to obtain the final prediction score. In this work, we follow [382] and adapt VBPR to multimodality by concatenating the visual and textual item features to generate a unique multimodal representation:

$$\hat{x}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i + \mathbf{f}_u^\top t(\mathbf{f}_i) \quad \text{with} \quad \mathbf{f}_i = \big\Vert_{m \in \mathcal{M}} \mathbf{f}_i^m, \tag{7.1}$$

where t is a projection function such that the latent dimensions of the multimodal user and item embeddings match.

MMGCN

One of the first approaches leveraging the representational power of graph convolutional networks (GCNs) with multimodal content is multimodal graph convolution network for recommendation [339] (dubbed as MMGCN). By designing one GCN for each modality, the model learns the different preferences users have towards each representation of the items. Finally, to fuse all multimodal representations into one for both users and items embeddings, the authors adopt the element-wise addition, and the predicted interaction score is calculated via the dot product:

$$\hat{x}_{ui} = \mathbf{f}_u^\top \mathbf{f}_i \quad \text{with} \quad \mathbf{f}_u = \sum_{m \in \mathcal{M}} c(\mathbf{e}_u, g(\mathbf{f}_u^m), t(\mathbf{f}_u^m, \mathbf{e}_u)), \tag{7.2}$$

where c and g are a combination and GCN-based functions. We report only the user-side formulation for the sake of space.

GRCN

Similarly to MMGCN, graph-refined convolutional network for multimedia recommendation [338] (dubbed as GRCN) utilizes a GCN-architecture to update user and item embeddings. Specifically, the adjacency matrix entries are refined by pruning the noisy user-item interactions according to the preference of users toward each item's modality. Collaborative and multimodal versions of the user and item embeddings are eventually combined through concatenation to estimate the interaction score via their dot product:

$$\hat{x}_{ui} = \mathbf{f}_u^\top \mathbf{f}_i \quad \text{with} \quad \mathbf{f}_u = g(\mathbf{e}_u, \mathbf{f}_u^m, \forall m \in \mathcal{M}) \,||\, \left(\underset{m \in \mathcal{M}}{||} t(\mathbf{f}_u^m) \right). \tag{7.3}$$

Again, we report only the user-wise formulation for lack of space.

LATTICE

Latent structure mining method for multimodal recommendation [382] (dubbed as LATTICE) performs graph structure learning on multiple modality-aware item-item graphs (one for each modality). The obtained adjacency matrices are aggregated through weighted element-wise addition, and the final adjacency matrix is exploited to perform graph convolution to update the representation of the collaborative item embeddings. Then, this updated version is added to the initial collaborative item embedding. Finally, the dot product between the collaborative user and (updated) item embeddings predicts the interaction score:

$$\hat{x}_{ui} = \mathbf{e}_u^\top \mathbf{f}_i \quad \text{with} \quad \mathbf{f}_i = \mathbf{e}_i + \frac{g(\mathbf{e}_i, \mathbf{f}_i^m, \forall m \in \mathcal{M})}{||g(\mathbf{e}_i, \mathbf{f}_i^m, \forall m \in \mathcal{M})||_2}, \tag{7.4}$$

where g is a LightGCN [126] architecture performing graph structure learning.

7.2.3 Proposed analysis

In this section, we present the details to conduct our analysis. Initially, we report on the used datasets, describing the methodologies employed for extracting multimodal features. Subsequently, we introduce and formally define the evaluation metrics employed, encompassing accuracy, diversity, and popularity bias. Finally, we provide a thorough summary of the reproducibility information for our study, detailing the methods used for dataset splitting and filtering as well as the strategy for hyperparameter search.

Datasets

The multimodal recommender systems have been tested on three popular [66, 155, 382, 407] datasets from the Amazon catalog [218]: Office Products (*Office*), (b) Toys & Games (*Toys*), and (c) Clothing, Shoes & Jewelry (*Clothing*). The multimodal datasets provide both images and descriptions for each available item. Specifically, we utilize the pre-extracted 4,096-dimensional visual features [82] which are made publicly available[2]. For the textual modality, we follow the existing literature [382], which aggregates the item's title, descriptions, categories, and brand, thereby generating textual embeddings by leveraging sentence transformers [256]. The generated features are 1,024-dimensional embeddings. Additional dataset information can be found in Table 7.7.

Evaluation metrics

In the proposed study, we refer to various metrics that may bring out additional insights which have not been investigated yet in multimodal recommendation. Indeed, we do not solely rely on accuracy metrics (i.e., Recall and nDCG) but also on diversity (i.e., iCov) and popularity bias (i.e., APLT) metrics. An ideal recommender system should increase all the metrics listed above according to the principle "higher is better" to boost accuracy and diversity while reducing the popularity bias of the produced recommendations. Nevertheless, with the current work, we try to unveil whether and why multimodal-aware recommender systems are affected by popularity bias. Thus, in the following, we will take into account those settings in which accuracy is high, while diversity and popularity bias are low (according to the metrics definitions).

Reproducibility

We investigate the models' behavior in three different settings: (i) *visual* modality, in which we employ only visual features, (ii) *textual* modality, in which we employ only textual features, and (iii) *multimodal*, where both modalities are combined.

In the first step, we evaluate the models in the multimodal setting which is the same setting as the original one for each tested approach. Then, we focused on quantifying the singular modality influence on the multimodal scenario in terms of accuracy, diversity, and popularity bias. Furthermore, to ensure the reproducibility of our work, in the following, we provide comprehensive details regarding the preprocessing and splitting of the datasets, as well as the tuning and evaluation of the models.

The datasets are filtered using the p-core strategy, where we set p to 5. Subsequently, we employ an 80%/20% train-test hold-out strategy to split the dataset. During the hyper-parameter tuning phase, we further divide the test set by removing 50% of its instances for the validation, specifically evaluating the results using the Recall@20 metric (as in the original work). In terms of models' training, we set the maximum

[2]https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html.

Table 7.5 Statistics of the tested datasets.

Datasets	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{R}\rvert$	**Sparsity (%)**
Office	4,905	2,420	53,258	99.5513
Toys	19,412	11,924	167,597	99.9276
Clothing	39,387	23,033	278,677	99.9693

number of epochs to 200 and select the model weights based on the epoch that yields the best performance on the validation set.

The code is implemented in Elliot [11]. Note that the explored hyper-parameter values are not entirely aligned with the ones in the original papers and codes. Indeed, we want to tune the selected baselines on an extensive, shared set of hyper-parameter values across all models for the sake of fair comparison.

7.2.4 Results and discussion

In this section, we answer the following research questions (RQs):

RQ1. *How do the selected multimodal-aware recommendation models behave in terms of accuracy, diversity, and popularity bias?* We investigate the recommendation performance in terms of accuracy (i.e., Recall, nDCG), diversity (i.e., iCov), and popularity bias (i.e., APLT). Note that, for the sake of completeness, we also report the performance of a recommender system generating recommendations in a random manner (i.e., Random) or based upon the most popular items in the catalog (i.e., MostPop); then, we train and evaluate MFBPR, that is the building model of the other multimodal baselines. We regard the performance of Random, MostPop, and MFBPR as a reference for the other multimodal-aware recommender systems we want to analyze.

RQ2. *What is the influence of each modality setting (i.e., visual, textual, multimodal) on such performance measures?* We take a step further by analyzing how each modality (i.e., visual, textual, and multimodal) influences accuracy, diversity, and popularity bias; the evaluation is conducted both on the single metric and across pairs of metrics.

Recommendation accuracy, diversity, and popularity bias

The results of the accuracy, diversity, and popularity bias metrics are reported in Table 7.6. The measured values refer to top@10, top@20, and top@50 recommendation lists. In the following, we discuss the obtained results considering the three metrics families separately.

Accuracy. Overall, LATTICE is the top-performing model, in alignment with the recent literature [382]. Indeed, its ability to learn more refined items' embeddings based upon the multimodal item-item similarities may positively impact the accuracy performance. Conversely, VBPR's outstanding performance to the other multimodal approaches comes as quite a surprise, considering that more complex and recent models leveraging GNNs (such as MMGCN and GRCN) do not outperform it.

Considering the performance on a dataset level, the most significant variation in metrics between LATTICE and VBPR is observed on *Toys* and *Clothing*, while the difference is reduced on *Office*. Notably, *Toys* and *Clothing* store three and four times more interactions than *Office*, respectively, but they are much sparser. This emphasizes LATTICE's ability to recommend more accurate items despite the higher dataset sparsity. Assessing the other models' performance, MMGCN works exceptionally well on *Toys* but shows the lowest performance as the number of interactions and sparsity increase. GRCN, in contrast, excels with highly sparse data, exhibiting an opposite trend to MMGCN.

From a metric-wise analysis, LATTICE outperforms VBPR in correctly predicting relevant items (high Recall) that are more likely to appear at the top of the recommendation lists (nDCG). However, the same trend is not as evident on the Recall, partly due to its normalization w.r.t. the k recommended items, which can lead to a smaller difference between LATTICE and VBPR as k increases.

Diversity. As far as recommendation diversity (i.e., iCov) is concerned, the worst-performing model is MMGCN, since its iCov is, in any case, negatively out of scale compared to the other models. For instance, when taking into account *Office*, MMGCN's iCov is slightly better than MostPop (whose item diversity is, by construction, the lowest) demonstrating a restricted ability to engage diverse items in the recommendation lists. Unexpectedly, the second-worst model is LATTICE, even if its performance is still more balanced to the other approaches than MMGCN's one. Indeed, we observe that while MMGCN is affected by poor accuracy due to the lack of item diversity, LATTICE can deal with both accuracy and diversity.

As an opposite (but noteworthy) trend, we underline that VBPR and GRCN are generally capable of recommending a wider portion of items than MMGCN and LATTICE, independently on the selected top-k. Overall, their iCov values are quite comparable to the ones of Random, which should provide (by definition) the highest

Table 7.6 Results in terms of recommendation accuracy (Recall, nDCG), diversity (iCov) and popularity bias (APLT). For accuracy metrics, ↑ means better performance, while ↓ means less diversity and more popularity bias. We remind that, while iCov and APLT metrics would generally adhere to the principle of "higher is better" (↑) for an ideal recommender system, in this work we consider the opposite as we want to emphasize which models are performing worst in terms of diversity and popularity bias.

Datasets	**Models**	top@10				top@20				top@50			
		Recall↑	**nDCG↑**	**iCov↓**	**APLT↓**	**Recall↑**	**nDCG↑**	**iCov↓**	**APLT↓**	**Recall↑**	**nDCG↑**	**iCov↓**	**APLT↓**
Office	Random	0.0034	0.0020	2,414	0.5950	0.0079	0.0034	2,414	0.5948	0.0220	0.0068	2,414	0.5924
	MostPop	0.0302	0.0208	20	0.0000	0.0533	0.0282	32	0.0000	0.1143	0.0439	66	0.0000
	MFBPR	0.0602	0.0389	2,268	0.2294	0.0955	0.0500	2,357	0.2379	0.1657	0.0677	2,398	0.2513
	VBPR	0.0652	0.0419	2,265	0.2321	0.1025	0.0533	2,354	0.2375	0.1774	0.0721	2,404	0.2469
	MMGCN	0.0455	0.0300	**74**	**0.0016**	0.0798	0.0405	**112**	**0.0078**	0.1575	0.0598	**247**	**0.0205**
	GRCN	0.0393	0.0253	2,390	0.3438	0.0667	0.0339	2,409	0.3469	0.1250	0.0488	2,414	0.3548
	LATTICE	**0.0664**	**0.0449**	2,121	0.1752	**0.1029**	**0.0566**	2,315	0.2039	**0.1780**	**0.0751**	2,397	0.2413
Toys	Random	0.0011	0.0006	11,879	0.4894	0.0021	0.0008	11,879	0.4896	0.0051	0.0015	11,879	0.4902
	MostPop	0.0130	0.0075	13	0.0000	0.0229	0.0104	24	0.0000	0.0451	0.0156	56	0.0000
	MFBPR	0.0641	0.0403	10,016	0.1167	0.0903	0.0481	10,944	0.1268	0.1394	0.0596	11,544	0.1460
	VBPR	0.0710	0.0458	10,085	0.1064	0.1006	0.0545	11,026	0.1180	0.1523	0.0667	11,624	0.1400
	MMGCN	0.0256	0.0150	**4,499**	0.0961	0.0426	0.0200	**6,238**	0.1058	0.0785	0.0285	**8,657**	0.1263
	GRCN	0.0554	0.0354	11,007	0.2368	0.0831	0.0436	11,609	0.2482	0.1355	0.0559	11,847	0.2679
	LATTICE	**0.0805**	**0.0512**	8,767	**0.0546**	**0.1165**	**0.0617**	10,285	**0.0684**	**0.1771**	**0.0759**	11,397	**0.0950**
Clothing	Random	0.0004	0.0002	23,016	0.4487	0.0010	0.0003	23,016	0.4478	0.0024	0.0006	23,016	0.4482
	MostPop	0.0089	0.0046	13	0.0000	0.0157	0.0063	24	0.0000	0.0322	0.0095	56	0.0000
	MFBPR	0.0303	0.0156	18,414	0.0729	0.0459	0.0195	20,582	0.0824	0.0734	0.0249	22,171	0.1017
	VBPR	0.0339	0.0181	19,195	0.0809	0.0529	0.0229	21,251	0.0915	0.0847	0.0292	22,555	0.1112
	MMGCN	0.0227	0.0119	**1,744**	**0.0044**	0.0348	0.0150	**2,864**	**0.0066**	0.0609	0.0201	**5,373**	**0.0121**
	GRCN	0.0319	0.0164	21,490	0.2358	0.0496	0.0209	22,503	0.2459	0.0858	0.0281	22,954	0.2631
	LATTICE	**0.0502**	**0.0275**	13,463	0.0134	**0.0744**	**0.0336**	17,538	0.0207	**0.1186**	**0.0425**	21,458	0.0385

item coverage from the catalog. We intend to further investigate (and justify) this aspect by assessing the effects of popularity bias.

Popularity bias. In terms of popularity bias (i.e., APLT), the worst and second-worst models are once again MMGCN and LATTICE (the former on *Office* and *Clothing*, while the latter on *Toys*). As already discussed, it makes sense to conceptually bind iCov and APLT. When assessing MMGCN's performance on *Office*, it becomes clear how the model is recommending only a few items (see again the iCov) while achieving good results in terms of accuracy; this demonstrates how the user-item interactions from *Office* may likely be biased towards popular items, and the phenomenon is even amplified due to the dataset small size. The same does not hold on *Clothing* where MMGCN, usually prone to popularity bias, gets also really low performance in terms of accuracy. Conversely, LATTICE can recommend popular items thus pushing its accuracy performance without amplifying the popularity bias phenomenon as much as MMGCN does. Indeed, even if LATTICE's iCov is the second-worst across all the datasets, the metric is always close to the best models in terms of diversity.

Finally, VBPR and GRCN confirm their ability (already observed on the diversity measure) to tackle also popularity bias in all experimental settings. Particularly, while we recognize that VBPR performance is slightly increased with respect to MFBPR in terms of iCov and APLT (the two approaches are almost similar), GRCN results are quite remarkable. It might be the case that its graph edges pruning technique (driven by multimodal signals) is reducing the influence of noisy user-item interactions (i.e., redundant edges which might involve popular items), thus helping to diversify the recommendations by considering also several long-tail items.

Summary. *In a standard multimodal setting, LATTICE stands out for its accuracy performance and ability to handle dataset sparsity, but at the detriment of amplifying popularity bias; MMGCN struggles with diversity, exhibits strong popularity bias, and sacrifices accuracy in certain scenarios; VBPR and GRCN, in different manners, better manage all the metrics by finding the right compromise among them.*

Modalities influence on recommendation performance

While the previous section has answered how multimodal recommender systems perform in terms of accuracy, diversity, and popularity bias when leveraging the ***full*** modalities, in the following, we discuss the influence of each ***single*** modality on the performance. We consider two evaluation dimensions where modalities influence is assessed (i) on accuracy, diversity, and popularity bias separately, and (ii) on pairs of metrics to investigate their joint variations.

Modalities influence on the single metric. Figure 7.2 displays the influence of each modality calculated as percentage variation with respect to the multimodal baseline, on the top@20 recommendation lists. We select the Recall (Figure 7.2a), iCov (Figure 7.2b), and APLT (Figure 7.2c) for accuracy, diversity, and popularity bias.

As regards the accuracy performance (Figure 7.2a), we notice how the trend is not consistent across all the datasets and models. Particularly, when considering *Office*, we observe that only VBPR and LATTICE fully exploit multimodality (indeed, their performance decreases when the modalities are injected separately); on an opposite level, on MMGCN, the visual modality slightly improves the multimodal setting, while the textual modality even worsens it; then, GRCN achieves better performance on both the visual and textual modalities, suggesting that this approach may not take advantage of the multimodal configuration. On the *Toys* dataset, the only textual setting generally improves the performance, bringing important information to the model learning interaction. The model benefiting from the single modality the most is MMGCN, which has an improvement of at least 20% on both visual and textual. For the remaining models, the trend is quite stable with the textual and visual modalities improving and reducing the performance, respectively. Finally, we observe that *Clothing* is the only dataset showing consistent trends. Indeed, the visual modality reduces

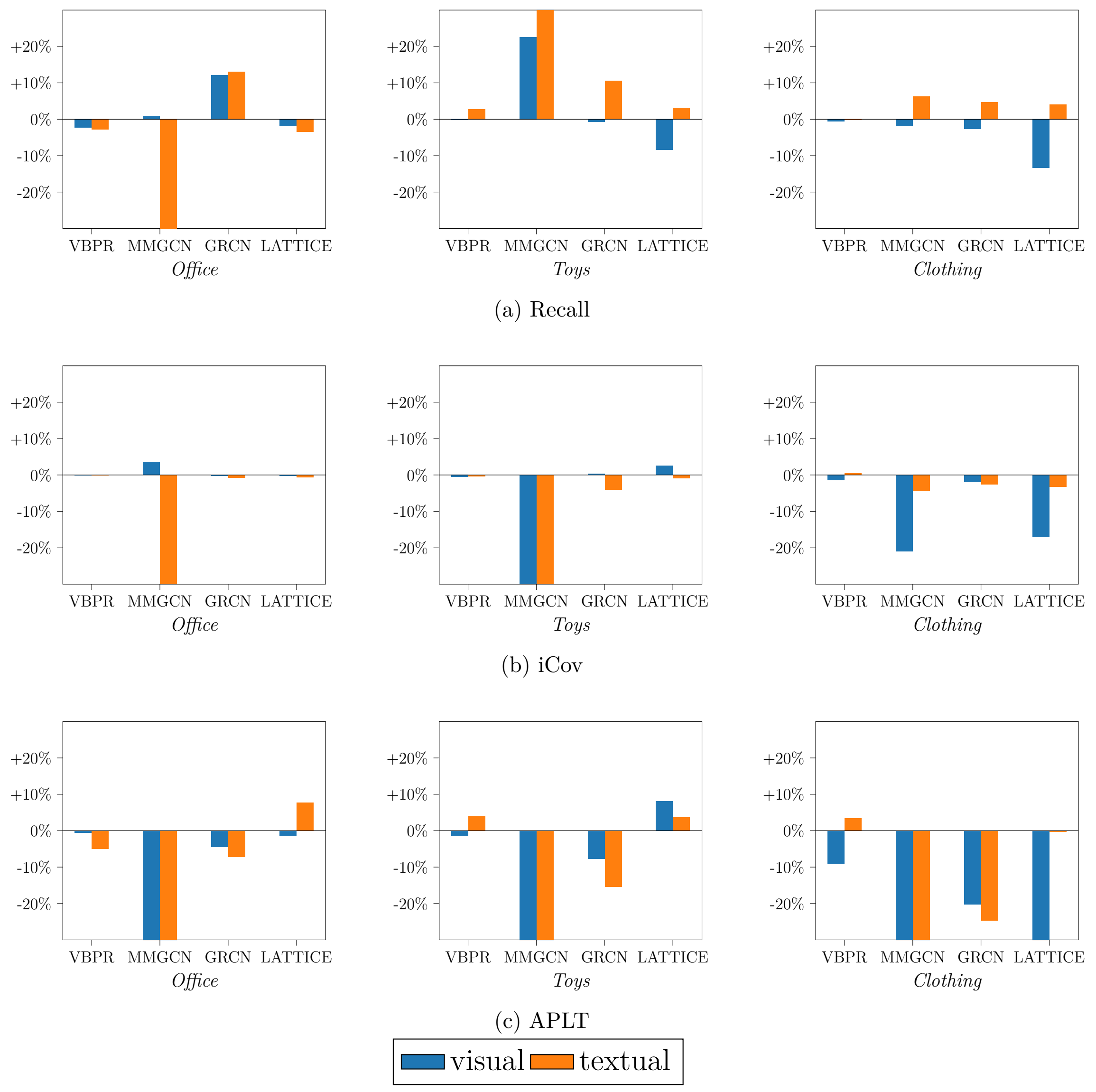


Fig. 7.2 Percentage variation on the (a) Recall, (b) iCov, and (c) APLT when training the multimodal recommender systems with either visual or textual modalities. The 0% line stands for the reference performance provided by the multimodal version of the model. All results refer to the top@20 recommendation lists.

the Recall while the textual increases it (with the only exception of VBPR whose percentage variation is negligible).

Differently from the accuracy analysis, we recognize a quasi-stable trend in the performance variation measured for the diversity metric (Figure 7.2b). Considering the *Office* dataset, each modality's contribution is generally irrelevant except for

MMGCN, for which the visual modality slightly improves the coverage across the whole recommendation list, while the textual one worsens the performance by a large margin. Assessing the trend on *Toys*, both the modalities decrease the coverage performance of the model when injected separately in the recommendation pipeline; remarkably, MMGCN is once again the model affected by the single modality presence the most, but this time the coverage performance widely deteriorates because of both the visual and textual modalities. Finally, on *Clothing*, both modalities lower the model's item coverage, with specific reference to the visual modality.

As the last part of our analysis, we take into account each modality's contribution to the popularity bias dimension (Figure 7.2c). Starting from *Office*, we notice how both modalities are prone to enforce popularity bias if injected singularly, with the only exception of LATTICE whose textual modality limits the popularity bias (the APLT increases); this is interesting as we remind that LATTICE is the second-worst model in terms of popularity bias, but using only the textual modality reduces its accuracy performance and the influence of popular items in the recommendation list. When it comes to the *Toys* dataset, every single modality enforces the popularity bias of MMGCN and GRCN; for VBPR, the visual and textual modalities amplify and reduce the bias, respectively, while for LATTICE both the visual and textual modalities limit the popularity bias. Finally, on *Clothing*, both the modalities show to increase the popularity bias of the model (but the textual one on VBPR and LATTICE).

Modalities cross-influence on metrics pairs. To conclude, we discuss the cross-influence of each modality setting (i.e., visual, textual, and multimodal) on pairs of metrics. In this respect, we decide to display (Figure 7.3) the joint trend of (a) accuracy and popularity bias (i.e., Recall *vs.* APLT), (b) accuracy and diversity (i.e., Recall *vs.* iCov), and (c) diversity and popularity bias (i.e., iCov *vs.* APLT). We only report the results on *Clothing* for top@20 recommendations.

In detail, VBPR and MMGCN are the models being affected by each specific modality the least, since the performance measures assessed on visual and textual are generally aligned with the multimodal reference. Regarding LATTICE, we notice that the textual modality has a major accuracy influence with respect to popularity bias and diversity. Indeed, the textual modality improves the Recall without having a relevant effect in terms of iCov and APLT; conversely, the visual modality reduces the accuracy by jointly worsening the diversity and the popularity bias. Finally, when considering GRCN, we observe that the multimodal setting reduces the popularity bias without affecting the accuracy and diversity.

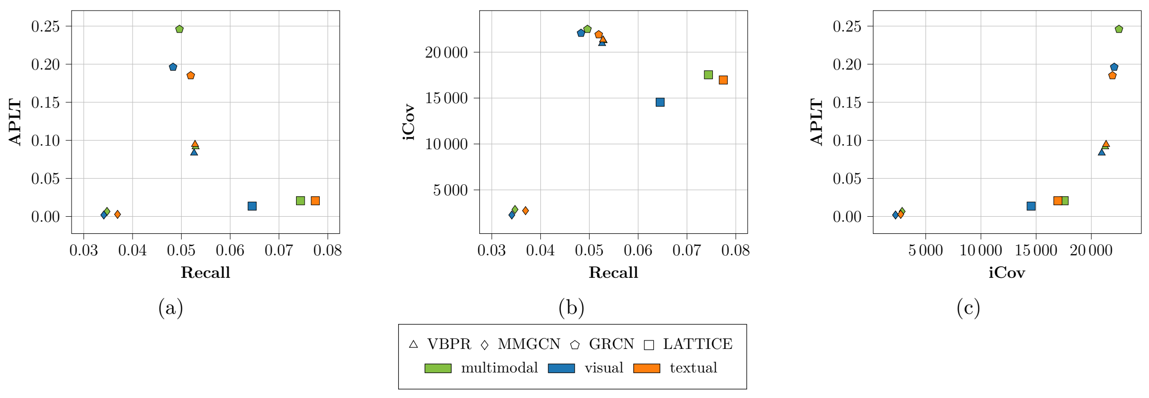


Fig. 7.3 Performance analysis on *Clothing* when comparing (a) Recall *vs.* APLT, (b) Recall *vs.* iCov, and (c) iCov *vs.* APLT for different modality settings involving the multimodal, visual, and textual modalities. Metrics are on top@20.

Summary. *In a single modality setting, the textual one improves the accuracy, while both modalities negatively affect the diversity and reinforce the popularity bias. When evaluating the modalities' influence across metrics pairs, the textual modality has a significant influence on accuracy but minimal effects on diversity and popularity bias; conversely, the visual modality reduces accuracy and jointly worsens the popularity bias and diversity.*

7.3 A comprehensive benchmarking within Elliot

After a separate evaluation of multimodal-aware recommender systems under accuracy, novelty, diversity, and popularity bias measures, we devote this section of the chapter to a comprehensive analysis across all recommendation metrics, with an extensive set of datasets and models. Specifically, we benchmark six state-of-the-art multimedia recommendation approaches (i.e., VBPR [125], MMGCN [339], GRCN [338], LATTICE [382], BM3 [407], and FREEDOM [406]). In the following, we report on the datasets we used, the technical aspects of the multimodal-aware recommender systems involved, the evaluation metrics we adopted (spanning both accuracy and beyond-accuracy recommendation metrics), the reproducibility details of our framework, and the results of the benchmarking analysis.

Table 7.7 Statistics of the tested datasets.

Datasets	$\|\mathcal{U}\|$	$\|\mathcal{I}\|$	$\|\mathcal{R}\|$	**Sparsity (%)**
Office	4,905	2,420	53,258	99.55%
Toys	19,412	11,924	167,597	99.93%
Beauty	22,363	12,101	198,502	99.93%
Sports	35,598	18,357	296,337	99.95%
Clothing	39,387	23,033	278,677	99.97%

7.3.1 Datasets

For the benchmarking, we use five popular [66, 155, 382, 407] datasets which collect the purchase history from five product categories of the Amazon catalog [124, 218], namely, Office Products (i.e., Office), Toys & Games (i.e., Toys), All Beauty (i.e., Beauty), Sports & Outdoors (i.e., Sports), and Clothing Shoes & Jewelry (i.e., Clothing). Besides containing the records of user-product interactions with timestamps and other metadata, such datasets come with the visual features extracted from the product images, stored as 4,096-dimensional embeddings which are publicly available at the same URL of the datasets[3]. Conversely, in terms of textual modality, we adopt the same procedure indicated in [382], and concatenate the item's title, descriptions, categories, and brand, to extract the 1,024-dimensional textual embeddings through sentence transformers [256]. Overall dataset information can be found in Table 7.7.

7.3.2 Multimedia recommender systems

We decide to benchmark the results of six state-of-the-art multimedia recommender systems, namely, VBPR [125], MMGCN [339], GRCN [338], LATTICE [382], BM3 [407], and FREEDOM [406]. Such approaches represent a group of techniques that are widely recognized in the related literature as strong baselines in multimedia recommendation exploiting multimodality, as well as recently proposed solutions at top-tier conferences (Table 7.8). In the following, we only focus on BM3 and FREEDOM, as the other multimodal-aware recommendation systems have been widely presented in previous sections of this chapter. Note that, by adding BM3 and FREEDOM to the set of benchmarked baselines, we provide an extensive analysis on five state-of-the-art graph-based approaches leveraging multimodal content.

[3]https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html.

Table 7.8 An overview on the selected multimedia recommender systems, along with their publication venue and year, and a non-exhaustive set of papers where they are used as baselines.

Models	Year	Venue	Baseline in
VBPR [125]	2016	AAAI	[66, 69, 181, 187, 305, 382]
MMGCN [339]	2019	MM	[293, 320, 336, 337, 380, 407]
GRCN [338]	2020	MM	[170, 186, 187, 336, 382, 407]
LATTICE [382]	2021	MM	[155, 223, 336, 383, 407]
BM3 [407]	2023	WWW	[180, 372, 404]
FREEDOM [406]	2023	MM	[404]

BM3

Bootstrapped multimodal model [407], indicated as BM3, proposes a self-supervised multimodal technique for recommendation. Different from previous approaches using computationally expensive augmentations, BM3 leverages dropout as a simple operation for generating contrastive views of the same embeddings. In detail, the loss function consists of three components, where a reconstruction loss minimizes the similarity between the contrastive views of user and item embeddings, while an inter- and intra-modality alignment loss works to minimize the distance between the contrastive views generated for the same or different modalities.

FREEDOM

The authors from [406] demonstrate that freezing the item-item multimodal similarity graph (derived from LATTICE) and denoising the user-item graph can lead to improved recommendation performance (the proposed model is named FREEDOM). As for the denoising operation of the user-item graph, the authors propose a degree-sensitive edge pruning to remove noisy edges from the user-item adjacency matrix. Moreover, and differently from LATTICE, the model optimizes a double BPR-like loss function, where the first component of the loss integrates a multimodal-enhanced representation of the item embedding, while the second component explicitly leverages the item projected multimodal features.

7.3.3 Evaluation metrics

To conduct the benchmarking analysis, we measure the recommendation performance through accuracy and beyond-accuracy metrics. For the recommendation accuracy, we consider the Recall@k and the nDCG@k; for the novelty [311] and diversity [295], we measure the EFD@k and the Gini@k, respectively; for the popularity bias [3], we calculate the APTL@k; finally, as a general index of how recommendations cover the entire catalog of products, we adopt the iCov@k.

7.3.4 Reproducibility

First, we pre-process the datasets following the 5-core filtering on users and items to remove cold-start users and items as done in [382]. Second, we split them according to the 80:20 hold-out strategy for the training and test sets, where the former and the latter contain the 80% and the 20% of interactions recorded for each user, respectively. Then, we decide to train the recommendation models so that the number of epochs (i.e., 200) and the batch size (i.e., 1024) are the same for all of them to ensure fair comparison. As for the other models' hyper-parameters, we follow a grid search strategy with 10 explorations comprising both the learning rate and the regularization coefficients and fix the remaining (model-specific) hyper-parameters to the best values according to the original papers and/or codes. Finally, to select the best configuration for each model and dataset, we remove the 50% of the test set for the validation set (following again [382]), and select the hyper-parameter setting providing the highest Recall@20 value on the validation data measured for a specific epoch (maximum 200 epochs). To foster the reproducibility of the proposed benchmarks, we provide the codes, datasets, and configuration files to replicate our results at: https://github.com/sisinflab/Formal-MultiMod-Rec, where we integrated the selected multimedia recommender systems into Elliot [11].

7.3.5 Benchmarking results

Table 7.9 reports on the results of the extensive benchmarking analysis we conduct on the selected datasets and state-of-the-art multimedia recommendation systems. The calculated metrics involve both accuracy (i.e., Recall and nDCG) and beyond-accuracy (i.e., EFD, Gini, APLT, and iCov) measures when considering top@10 and top@20 recommendation lists. Based on how we defined all the recommendation metrics, higher values indicate better performance.

In terms of **accuracy** performance, we observe that one of LATTICE, BM3, and FREEDOM is steadily among the two best recommendation models, which is something that also emerges from the related literature. This holds across all datasets and top@k under analysis. Nevertheless, we notice an interesting trend when considering VBPR. Indeed, we observe how this model is quite always among the top-3 recommendation techniques despite being one of the shallowest approaches compared to other more recent and complex models. As already stated in recent works [206, 207], this finding demonstrates how even a not-so-deep, but still careful hyper-parameter exploration (such as the one we performed) may help uncover unexpected results with respect to what described in the literature.

However, the most surprising behavior involves the analysis of the **beyond-accuracy** performance. While several works depict the most recent approaches in multimedia recommendation (i.e., LATTICE, BM3, and FREEDOM) as dominating the accuracy level, the same does not hold for other metrics accounting for novelty, diversity, and popularity bias. Indeed, the only observable trend in this setting is that GRCN and VBPR steadily settle as best-performing algorithms. Particularly, it is worth pointing out how both approaches can strike a sufficient trade-off between accuracy and beyond-accuracy measures, where VBPR can even reach quite high performance on beyond-accuracy without giving up on accuracy that much. Once again, such observations corroborate what has recently been pointed out in similar works [206, 207], by extending the analysis to additional datasets and multimedia recommendation systems.

7.4 Leveraging textual review content on graph edges for recommendation

The message-passing pattern, as used in graph-based recommendation, may still present some limitations despite being successful. An argument could be made that not all user-item interactions (i.e., graph edges) have the same relative importance. To clarify this, consider the motivating scenario in Figure 7.4, where we depict a subset of users and items from a real-world e-commerce platform (i.e., the Amazon catalog) and enrich their interactions with ratings and reviews. Both user u_1 and u_2 interacted with item i_1, thus inferring that they might share similar interests and preferences. However, careful analysis of the corresponding reviews reveals that their **opinions** about item i_1 are opposite (the expressed ratings are 5 and 2, respectively). Following a similar reasoning schema, users u_1 and u_3 have both interacted with item i_2 but their **comments**, while being generally similar (the item is rated 3 and 5, respectively), show slight shades of disagreement (i.e., u_1 is not completely satisfied with the belt size). As the message-passing pattern works by indiscriminately aggregating the neighbor nodes **at multiple hops**, the node representation of u_1 is ultimately influenced by the

Table 7.9 Benchmarking results on selected datasets and state-of-the-art multimedia recommender systems. The reported values refer to accuracy and beyond-accuracy recommendation metrics, on top@10 and top@20 recommendation lists. For each metric-dataset pair, **boldface** and underline indicate best and second-to-best values.

Datasets	**Models**	top@10						top@20					
		Accuracy		**Beyond-accuracy**				**Accuracy**		**Beyond-accuracy**			
		Recall	**nDCG**	**EFD**	**Gini**	**APLT**	**iCov**	**Recall**	**nDCG**	**EFD**	**Gini**	**APLT**	**iCov**
Office	VBPR	0.0652	0.0419	0.1753	0.3634	0.2321	93.83%	0.1025	0.0533	0.1479	0.3960	0.2375	97.51%
	MMGCN	0.0455	0.0300	0.1140	0.0128	0.0016	3.07%	0.0798	0.0405	0.1027	0.0231	0.0078	4.64%
	GRCN	0.0393	0.0253	0.1215	**0.4587**	**0.3438**	**99.01%**	0.0667	0.0339	0.1051	**0.4892**	**0.3469**	**99.79%**
	LATTICE	0.0664	0.0449	0.1827	0.2128	0.1752	87.86%	0.1029	0.0566	0.1513	0.2652	0.2039	95.90%
	BM3	**0.0701**	**0.0460**	**0.1837**	0.1407	0.1427	77.13%	**0.1081**	**0.0583**	**0.1550**	0.1900	0.1715	91.55%
	FREEDOM	0.0560	0.0365	0.1493	0.1922	0.1875	79.12%	0.0884	0.0469	0.1282	0.2439	0.2080	90.64%
Toys	VBPR	0.0710	0.0458	0.1948	0.2645	0.1064	84.90%	0.1006	0.0545	0.1527	0.3011	0.1180	92.82%
	MMGCN	0.0256	0.0150	0.0648	0.0989	0.0961	37.87%	0.0426	0.0200	0.0570	0.1450	0.1058	52.51%
	GRCN	0.0554	0.0354	0.1604	**0.3954**	**0.2368**	**92.66%**	0.0831	0.0436	0.1298	**0.4329**	**0.2482**	**97.73%**
	LATTICE	0.0805	0.0512	0.2090	0.1656	0.0546	73.80%	0.1165	0.0617	0.1665	0.2026	0.0684	86.58%
	BM3	0.0613	0.0393	0.1582	0.0776	0.0486	56.23%	0.0901	0.0478	0.1270	0.1154	0.0658	73.50%
	FREEDOM	**0.0870**	**0.0548**	**0.2284**	0.1474	0.0756	62.09%	**0.1249**	**0.0660**	**0.1820**	0.2007	0.0951	78.42%
Beauty	VBPR	0.0760	0.0483	0.2119	0.2076	0.0833	83.06%	0.1102	0.0586	0.1700	0.2376	0.0915	91.41%
	MMGCN	0.0496	0.0294	0.1300	0.0252	0.0282	13.75%	0.0772	0.0379	0.1105	0.0423	0.0345	21.37%
	GRCN	0.0575	0.0370	0.1817	**0.3823**	**0.2497**	**94.59%**	0.0892	0.0466	0.1498	**0.4178**	**0.2608**	**98.56%**
	LATTICE	**0.0867**	**0.0544**	0.2272	0.1153	0.0386	65.82%	0.1259	0.0661	0.1830	0.1558	0.0511	81.60%
	BM3	0.0713	0.0443	0.1831	0.0245	0.0179	32.31%	0.1051	0.0545	0.1490	0.0414	0.0228	48.75%
	FREEDOM	0.0864	0.0539	**0.2279**	0.0921	0.0486	55.89%	**0.1286**	**0.0666**	**0.1868**	0.1359	0.0653	72.96%
Sports	VBPR	0.0450	0.0281	0.1167	0.1501	0.0497	75.77%	0.0677	0.0349	0.0949	0.1722	0.0552	86.54%
	MMGCN	0.0342	0.0207	0.0791	0.0095	0.0046	5.10%	0.0551	0.0269	0.0678	0.0168	0.0065	8.39%
	GRCN	0.0330	0.0202	0.0885	**0.3087**	**0.2190**	**91.28%**	0.0523	0.0259	0.0746	**0.3386**	**0.2273**	**97.09%**
	LATTICE	**0.0610**	0.0372	0.1465	0.0573	0.0129	48.44%	0.0898	0.0456	0.1185	0.0802	0.0185	64.90%
	BM3	0.0548	0.0349	0.1372	0.0776	0.0283	59.13%	0.0825	0.0430	0.1118	0.1120	0.0385	76.75%
	FREEDOM	0.0603	**0.0375**	**0.1494**	0.0621	0.0319	48.37%	**0.0911**	**0.0465**	**0.1219**	0.0926	0.0442	65.81%
Clothing	VBPR	0.0339	0.0181	0.0502	0.2437	0.0809	83.40%	0.0529	0.0229	0.0413	0.2791	0.0915	92.33%
	MMGCN	0.0227	0.0119	0.0292	0.0136	0.0044	7.58%	0.0348	0.0150	0.0240	0.0236	0.0066	12.44%
	GRCN	0.0319	0.0164	0.0481	**0.3990**	**0.2358**	**93.37%**	0.0496	0.0209	0.0397	**0.4368**	**0.2459**	**97.77%**
	LATTICE	0.0502	0.0275	0.0738	0.1022	0.0134	58.49%	0.0744	0.0336	0.0589	0.1384	0.0207	76.20%
	BM3	0.0418	0.0226	0.0596	0.1348	0.0319	72.88%	0.0633	0.0281	0.0486	0.1825	0.0449	88.65%
	FREEDOM	**0.0547**	**0.0294**	**0.0805**	0.1509	0.0600	65.54%	**0.0822**	**0.0363**	**0.0652**	0.2078	0.0843	81.91%

representations of both u_2 and u_3 after two propagation hops. In the long term, such behavior may lead to what we could define as a **node representation error**.

Weighting the importance of neighborhood while aggregating the incoming messages into the ego node is among the prominent solutions to the abovementioned issue. Following the same direction path in [313], other popular and recent works in recommendation such as [328, 330, 351, 384] leverage attention mechanisms (i.e., a neural network) to perform the weighting procedure. Even if these models have widely demonstrated to provide superior accuracy recommendation performance, they are still affected by **oversmoothing**, the phenomenon according to which node embedded representations tend to get closer and closer in the latent space after multiple propagation hops, thus flattening the existing differences in the neighborhood [54, 405]. For this reason, attention-based approaches usually propagate messages for only one or two hops, but this does not help access wider portions of the user-item graph.

In this respect, we believe attention-based techniques generally disregard other potential sources of information (e.g., users' generated reviews) whose contribution may positively impact the neighborhood weighting process. Opinions and comments about interacted items constitute the basis on which like-minded users gather on online platforms, as they promote the discovery of novel and diverse items from the catalog. In the last part of this chapter, we first formally define the problem of nodes' representation error in graph collaborative filtering. After that, we show how existing weighting techniques (such as attention mechanisms) may alleviate the described issue at the expense of limiting the hop exploration depth to reduce the effect of oversmoothing. Thus, to address such drawback, we propose a lighter-weighting procedure that exploits the informative content extracted from reviews (i.e., opinions and comments about interacted items) to enhance graph edge representation. Such edge-enriched features are eventually used to derive the similarity between the ego node and its neighbors, which we re-interpret as the importance of the neighbor node on the ego node. Our proposed weighting procedure is applied to a GCN acting as the correction to another traditional (but error-affected) GCN. We call our solution EGCF, which stands for Edge Graph Collaborative Filtering.

After formalizing the theoretical basis for EGCF and its rationale, we assess its efficacy on three popular product categories from the Amazon catalog [228]. Given their similar intuitions and rationale to EGCF, we compare the method with four families of CF-based recommendation, i.e., traditional, review-based [62, 286], and graph-based approaches (both leveraging attention mechanisms and not). We seek to answer these research questions about our proposed approach:

- **RQ1.** Can the correction to the node error representation help EGCF produce more accurate recommendations than state-of-the-art baselines?
- **RQ2.** Considering the high impact that novel and diverse recommendation lists may have on both users and companies, how effective is EGCF when evaluated on beyond-accuracy metrics, given its strategy for neighborhood exploration?
- **RQ3.** What is the effect of changing the hop exploration number on recommendation performance, and how can we justify such behaviors for the adopted architecture?

The extensive experimental evaluation shows that the correction to the node representation error and the possibility of propagating messages across multiple hops permits EGCF to outperform state-of-the-art baselines on accuracy and beyond-

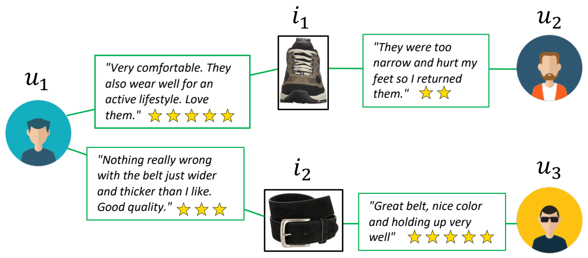

i_1
u_2
u_1
"Very comfortable. They also wear well for an active lifestyle. Love them." ★★★★★
"They were too narrow and hurt my feet so I returned them." ★★
"Nothing really wrong with the belt just wider and thicker than I like. Good quality." ★★★
i_2
u_3
"Great belt, nice color and holding up very well" ★★★★★

Fig. 7.4 A subset of users, items, and reviews users wrote about items, along with the expressed ratings (in the range 1-5). Despite being connected to the same items, users u_1-u_2, and users u_1-u_3 do not share similar opinions about the interacted items.

accuracy metrics. Finally, the study on the hop propagation number proves the soundness of our proposed architectural configuration.

7.4.1 Review-based recommendation

Reviews convey a rich source of information to access users' multi-faceted opinions about interacted items. For this reason, several existing works propose to extract valuable knowledge from them to produce better-tailored recommendations [62, 286]. Among the pioneer works, Wang et al. [318] adopt a stacked denoising autoencoder to approximate the user-item rating matrix starting from textual reviews, Almahairi et al. [9] introduce two neural network-based approaches built upon bag-of-words and recurrent neural networks, and Kim et al. [154] present convolutional matrix factorization (ConvMF), where a convolutional neural network is merged with probabilistic MF to learn the context of review documents.

Reviews are textual documents composed of words, which may further be grouped into sentences. To exploit such hierarchical structure, Zheng et al. [399] design a convolutional neural network on top of a factorization machine prediction model to extract from review's words a unique embedded representation for users and items. The adoption of attention mechanisms may help refine each review component's importance on the recommendation profile of users and items. In this respect, Liu et al. [183] improve the previous approach by weighting the importance of convolutionally-embedded reviews for both users and items for the sake of explanation. Similarly, Lu et al. [196] learn users' and items' attention features by exploring different review components such as words, sentences, and topics via a GRU-based network, while Liu et al. [183] (based upon the solution described in [184]) augment users' and items' collaborative latent factors through features extracted from their generated ratings and reviews. Wang et al. [323] leverage common review properties (e.g., how helpful the reviews were for other users) to assess its importance on users and items.

Only recently, very few works have injected the informative content of reviews into graph-based networks for recommendation. Wu et al. [340] propose a model named reviews meet graphs (RMG), a multi-view framework that learns users' and items' representation by considering the word- and sentence-level of reviews and exploring two hops of the user-item graphs to access also user-user and item-item relations. Gao et al. [101] present a three-structured architecture that catches the short- and long-term user preferences and item features, along with the collaborative information encoded in the bipartite user-item graph. Shi et al. [279] introduce a dual GCN model, where one extracts and propagates review aspects, and the other reuses the aspect for the graph.

Despite addressing recommendation through different strategies, the presented algorithms generally work by grouping reviews on both users and items profiles but, in fact, limiting the exploration of users and items neighbors at one hop (i.e., the nearest neighborhood). Conversely, our proposed approach exploits reviews as edge side information to describe user-item interactions and propagate their informative content at multiple hops to overcome theoretical issues in the way graph-based recommender systems are usually designed (see later).

7.4.2 Methodology

The section presents and motivates our proposed method, Edge Graph Collaborative Filtering (EGCF). We highlight a potentially critical issue in the message-passing schema for graph-based recommendation. Even if weighting the importance of each neighbor node may alleviate the problem, we discuss the insights and propose an enhanced application of the importance weighting.

Notation and preliminaries

Inspired by popular approaches [158], current graph-based recommender systems refine users' and items' node embeddings by exploring their multi-hop interconnections represented in the graph. Let u and i be the nodes for a user and an item to be updated (i.e., the ego nodes), and let $\mathcal{N}(u)$ and $\mathcal{N}(i)$ be the sets of nodes at one hop from u and i, respectively (i.e., their neighborhood). The ego node embeddings $\mathbf{e}_u$ and $\mathbf{e}_i$ are updated by aggregating their neighborhoods (i.e., messages):

$$\begin{aligned}
\mathbf{e}_u^{(1)} &= \omega\left(\{\mathbf{e}_i, \forall i \in \mathcal{N}(u)\}\right) \\
\mathbf{e}_i^{(1)} &= \omega\left(\{\mathbf{e}_u, \forall u \in \mathcal{N}(i)\}\right)
\end{aligned} \tag{7.5}$$

where $\mathbf{e}_u^{(1)}$ and $\mathbf{e}_i^{(1)}$ are the refined embedding versions of user u and item i after one hop, while $\omega(\cdot)$ indicates the aggregation function. This message-passing pattern may be iterated L times, thus exploring wider and wider neighborhoods of the ego nodes. After two hops, the refined embeddings of user u and item i are:

$$\begin{aligned}\mathbf{e}_u^{(2)} &= \omega\left(\left\{\mathbf{e}_i^{(1)}, \forall i \in \mathcal{N}(u)\right\}\right)\\ \mathbf{e}_i^{(2)} &= \omega\left(\left\{\mathbf{e}_u^{(1)}, \forall u \in \mathcal{N}(i)\right\}\right)\end{aligned} \tag{7.6}$$

A limitation in the message-passing

The user formulation in Equation (7.6) can be expanded through Equation (7.5):

$$\mathbf{e}_u^{(2)} = \omega\left(\left\{\omega\left(\left\{\mathbf{e}_{u'}, \forall u' \in \mathcal{N}(i)\setminus\{u\}\right\}\right), \forall i \in \mathcal{N}(u)\right\}\right) \tag{7.7}$$

What emerges is that, by propagating messages at two hops, the node embedding of user u is eventually refined through the contributions from other users who interacted with the same items as u. In other words, after two hops, **each user profile is influenced by the profiles of other users who rated the same items.**

Indeed, this assumption is aligned with the rationale behind collaborative filtering, i.e., similar users are likely to interact with the same items. However, not all user-item interactions (i.e., graph edges) may be equally important to the users and items involved. Thus, indiscriminately aggregating neighbor node embeddings into the ego node could, after multiple hops, harm the node updating process by bringing all contributions from the neighborhood, even the noisy ones. We interpret this as a **node representation error**, propagating with the exploration hops in the graph.

For this reason, contributions coming from each neighbor node are usually weighted before aggregating them into the ego nodes, modifying the presented formula:

$$\begin{aligned}\mathbf{e}_u^{(2)} = \omega\Big(\alpha_{i\to u}^{(2)}\Big\{\omega\Big(\Big\{\alpha_{u'\to i}^{(1)}\mathbf{e}_{u'},\\ \forall u' \in \mathcal{N}(i)\setminus\{u\}\Big\}\Big), \forall i \in \mathcal{N}(u)\Big\}\Big)\end{aligned} \tag{7.8}$$

where $\alpha_{j\to k}^{(l)}$ stands for the importance that the neighbor node j has on the ego node k after l hops. These weights are generally calculated by means of attention mechanisms, and depend on the embeddings of the neighbor and the ego nodes they refer to, e.g.,

$\alpha^{(l)}_{j\to k} = \varphi\Big(\mathbf{e}^{(l-1)}_j, \mathbf{e}^{(l-1)}_k\Big)$, where $\varphi(\cdot,\cdot)$ is a neural network:

$$\begin{aligned}\mathbf{e}^{(2)}_u = \omega\Bigg(&\overbrace{\varphi\Big(\mathbf{e}^{(1)}_i, \mathbf{e}^{(1)}_u\Big)}^{(\square)}\Big\{\omega\Big(\Big\{\overbrace{\varphi(\mathbf{e}_{u'}, \mathbf{e}_i)}^{(\triangle)}\mathbf{e}_{u'},\\ &\forall u' \in \mathcal{N}(i)\setminus\{u\}\Big\}\Big), \forall i \in \mathcal{N}(u)\Big\}\Bigg)\end{aligned} \tag{7.9}$$

that is, $\mathbf{e}^{(2)}_u$ depends on ($\square$) the importance each neighbor item node i has on the ego user node u after one hop, and ($\triangle$) the importance all users interacting with the same items as u have on their items. Note that ($\square$) may be further expanded:

$$\begin{aligned}\varphi\Big(\mathbf{e}^{(1)}_i, \mathbf{e}^{(1)}_u\Big) &= \varphi\Big(\omega\Big(\Big\{\alpha^{(1)}_{u'\to i}\mathbf{e}_{u'}, \forall u' \in \mathcal{N}(i)\setminus\{u\}\Big\}\Big),\\ &\qquad \omega\Big(\Big\{\alpha^{(1)}_{i'\to u}\mathbf{e}_{i'}, \forall i' \in \mathcal{N}(u)\setminus\{i\}\Big\}\Big)\Big)\\ &= \varphi\Big(\omega\Big(\Big\{\varphi(\mathbf{e}_{u'}, \mathbf{e}_i)\mathbf{e}_{u'}, \forall u' \in \mathcal{N}(i)\setminus\{u\}\Big\}\Big),\\ &\qquad \omega\Big(\Big\{\varphi(\mathbf{e}_{i'}, \mathbf{e}_u)\mathbf{e}_{i'}, \forall i' \in \mathcal{N}(u)\setminus\{i\}\Big\}\Big)\Big)\end{aligned} \tag{7.10}$$

When merging Equation (7.9) and Equation (7.10):

$$\begin{aligned}\mathbf{e}^{(2)}_u = \omega\Bigg(&\overbrace{\varphi\Big(\mathbf{e}^{(1)}_i, \mathbf{e}^{(1)}_u\Big)}^{\overbrace{\varphi(\mathbf{e}_{u'}, \mathbf{e}_i)}^{(\square)}\;\overbrace{\varphi(\mathbf{e}_{i'}, \mathbf{e}_u)}^{(\triangle)}}\Big\{\omega\Big(\Big\{\overbrace{\varphi(\mathbf{e}_{u'}, \mathbf{e}_i)}^{(\square)}\mathbf{e}_{u'},\\ &\forall u' \in \mathcal{N}(i)\setminus\{u\}\Big\}\Big), \forall i \in \mathcal{N}(u)\Big\}\Bigg)\end{aligned} \tag{7.11}$$

The node embedding for user u after two hops depends on ($\square$) the importance of all users interacting with the same items as u on those items, and ($\triangle$) the importance of all items interacted by u on user u. In other words, weighting the importance of each neighbor node on the ego node before the aggregation allows, after two propagation hops, to calculate **to what extent each user profile is influenced by the profiles of the other users who rated the same items**. Without loss of generality, a similar consideration could be made after a number of hops greater than two.

Enhancing neighborhood weighting through reviews

As known, graph-based models in machine learning are affected by oversmoothing [54, 405]. This phenomenon leads node embeddings, after multiple propagation hops, to become closer and closer in their representation in the latent space, eventually flattening their existing differences. As this behavior would profoundly weaken models' performance, exploration of the neighborhood generally tends to be constrained to very few hops (e.g., a maximum of two hops in attention-based weighting). **However, in recommendation scenarios, limiting the exploration of the user-item bipartite graph may represent an inconsistency to the idea of collaborative filtering**, where users are connected to share preferences and tastes for similar items.

Under this assumption, we believe the neighborhood weighting process could be further enhanced by exploiting other sources of information that are not usually taken into account. In the majority of popular online platforms for e-commerce (e.g., Amazon), **reviews** are fundamental tools to share **opinions** and **comments** about interacted items, as they convey the multi-faceted aspects that drove a user to interact with an item. Leveraging such side information on the connections existing among users and items in the bipartite graph (i.e., graph edges) can improve the learning of the importance weights by reducing the oversmoothing effect because each user/item node embedding is conditioned on the opinion conveyed by the review.

Let $\mathcal{W}_{ui} = \{w_1, w_2, \dots, w_R\}$ be the set of R words that compose the review written by user u about item i. After an initial tokenization step, the sets of tokens for $\mathcal{W}_{ui}$ is defined as $\mathcal{T}_{ui} = \{t_1, t_2, \dots, t_T\}$. Tokens are mapped to word embeddings, which are injected into an opinion-based model pretrained to predict the rating expressed by the user through specific terms in the review. While the output model carries the single information about the predicted review score, the activation of a hidden layer would unveil a richer source of textual features (i.e., an embedding) which drove the opinion-based model to predict that score. High-level features extracted from pretrained deep learning models can boost the recommendation performance of recommender systems leveraging items' side information (e.g., visual-based recommender systems [83, 125]). We deem these textual features to deserve a pivotal role in this weighting process.

Let $\mathbf{r}_{ui} \in \mathbb{R}^f$ be the textual embedding extracted from the review of user u about item i through the pretrained opinion-based model. First, we project $\mathbf{r}_{ui} \in \mathbb{R}^f$ to the same latent space as $\mathbf{e}_u \in \mathbb{R}^d$ and $\mathbf{e}_i \in \mathbb{R}^d$ with a one-layer neural network:

$$\mathbf{p}_{ui} = \text{LeakyReLU}\left(\mathbf{W}\mathbf{r}_{ui} + \mathbf{b}\right) \tag{7.12}$$

where $\mathbf{p}_{ui} \in \mathbb{R}^d$ is the projected review embedding, while $\mathbf{W} \in \mathbb{R}^{f \times d}$ and $\mathbf{b} \in \mathbb{R}^d$ are the projection matrix and the bias, respectively. We seek to retain only those textual features of review $\mathbf{r}_{ui}$ **which can be significant to later calculate the interdependence between this embedding and user/item ones**.

Then, we propose to enhance the neighborhood weighting procedure at hop l by conditioning the importance weights also on the projected embedding of the review connecting user u and item i. For instance, the importance of the neighbor item node i on the ego user node u after l hops is calculated as:

$$\alpha_{i \to u}^{(l)} = \varphi\left(\mathbf{e}_i^{(l-1)}, \mathbf{e}_u^{(l-1)}, \mathbf{p}_{ui}\right) \tag{7.13}$$

Note that, **since $\mathbf{p}_{ui}$ cannot increase the impact of the oversmoothing effect (because it is not dependent on the hop l), its usage in the importance weight formula becomes even more beneficial**. Let us focus on the weighting function $\varphi(\cdot,\cdot,\cdot)$. Many approaches from the literature propose to leverage attention mechanisms, usually implemented as a neural network trained in the downstream task to predict the importance of the neighbor node on the ego node. In our solution, we opt for a simplified and lightweight formulation that seeks to calculate the similarity between the neighbor and the ego nodes, **conditioned on the opinion embedding of the review connecting them**. Specifically:

$$\alpha_{i \to u}^{(l)} = \cos\left(\mathbf{e}_i^{(l-1)} \odot \mathbf{p}_{ui}, \mathbf{e}_u^{(l-1)} \odot \mathbf{p}_{ui}\right) \tag{7.14}$$

where $\odot$ is the element-wise multiplication, and $\cos(\cdot,\cdot)$ is the cosine similarity. Note that we suppress negative similarities to zero as such weights are usually non-negative. Multiplying both node embeddings by the review opinion embedding provides the interplay between each node feature and the opinion features, **thus producing a modified version of the node representation that conveys a richer source of information.** No trainable projection weight is learned in the presented formulation since the contribution of the review embedding is meaningful enough.

A double message-passing schema

The proposed neighborhood weighting procedure can help correct the representation error generated in the traditional message-passing schema. However, the idea is not to completely replace it, as several recent works from the literature have demonstrated its efficacy, especially in producing accurate recommendations [126]. The proposed approach involves a double message-passing schema, where two graph models are trained to refine **their own user/item node representations**. While the first one aggregates the contributions coming from the neighbor nodes into the ego nodes by weighting the neighborhood importance on the ego node **statically**, the second one aggregates the neighborhood's messages which are also weighted through the **opinion** embeddings from reviews.

We define the two graph convolutional networks as GCN_e (*error-affected*) and GCN_c (*correction*) and assign the node embeddings $\mathbf{e}_*$ to GCN_e, and the node embeddings $\mathbf{c}_*$ to GCN_c. As for the aggregation function, in both cases, we sum the weighted messages coming from the neighbor nodes. As such, the update of the user node embedding u after l hops is calculated as:

$$\begin{aligned}
\mathbf{e}_u^{(l)} &= \sum_{i \in \mathcal{N}(u)} \alpha_{i \to u} \mathbf{e}_i^{(l-1)} = \sum_{i \in \mathcal{N}(u)} \frac{\mathbf{e}_i^{(l-1)}}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}} \\
\mathbf{c}_u^{(l)} &= \sum_{i \in \mathcal{N}(u)} \alpha_{i \to u} \alpha_{i \to u}^{(l)} \mathbf{c}_i^{(l-1)} = \\
&= \sum_{i \in \mathcal{N}(u)} \frac{\cos\left(\mathbf{e}_i^{(l-1)} \odot \mathbf{p}_{ui}, \mathbf{e}_u^{(l-1)} \odot \mathbf{p}_{ui}\right)}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}} \mathbf{c}_i^{(l-1)}
\end{aligned} \tag{7.15}$$

Note that $\alpha_{i \to u}$ is static and only depends on the topology of the bipartite graph, while $\alpha_{i \to u}^{(l)}$ varies along with the exploration hop and depends on the embeddings of ego/neighbor nodes, and the opinion review embedding. After L propagation hops, the final embedding representation is obtained as:

$$\begin{aligned}
\bar{\mathbf{e}}_u &= \sum_{l=0}^{L} \frac{1}{1+l} \mathbf{e}_u^{(l)}, \quad \bar{\mathbf{e}}_i = \sum_{l=0}^{L} \frac{1}{1+l} \mathbf{e}_i^{(l)} \\
\bar{\mathbf{c}}_u &= \sum_{l=0}^{L} \frac{1}{1+l} \mathbf{c}_u^{(l)}, \quad \bar{\mathbf{c}}_i = \sum_{l=0}^{L} \frac{1}{1+l} \mathbf{c}_i^{(l)}
\end{aligned} \tag{7.16}$$

where we apply the scaling factor $1/(1+l)$ to further alleviate the oversmoothing problem. A schematic overview of the node refining algorithm proposed for EGCF is displayed in Figure 7.5.

Given the learned *error-affected* and *correction* embeddings from above, EGCF predicts if a user u may interact with item i through the following formulation:

$$\hat{R}_{ui} = \underbrace{\bar{\mathbf{e}}_u^\top \bar{\mathbf{e}}_i}_{\text{error-affected}} + \underbrace{\bar{\mathbf{c}}_u^\top \bar{\mathbf{c}}_i}_{\text{correction}} \tag{7.17}$$

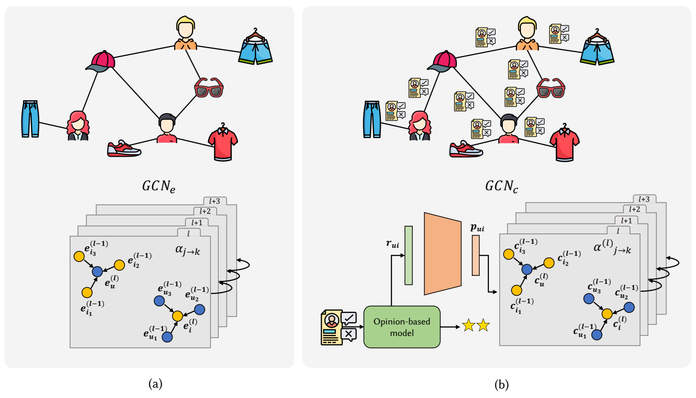


Fig. 7.5 Overview of the node refining algorithm proposed for EGCF. A statically-weighted GCN network affected by node representation error (a) is corrected through another GCN network (b), where an opinion-based embedding is extracted from each review as edge side information to weight the importance of the neighbor nodes on their ego nodes.

Thus, we apply the error correction to the user/item embedding representation only when predicting the user/item interaction. We optimize EGCF with the state-of-the-art Bayesian Personalized Ranking. (BPR) [258].

7.4.3 Experiments and discussion

Experimental setup

Datasets. We use three popular [66, 332] datasets from Amazon's Baby, Boys & Girls, and Men categories [228] which contain historical user-item interactions and reviews. We retain only interactions with non-empty reviews, then keep the 20k and 10k most popular items for Baby and Boys & Girls/Men, respectively. Finally, we apply the 5- and 15-core on items and users on Baby/Boys & Girls and Men, respectively. Statistics are in Table 7.10.
Baselines. We compare our approach with eight state-of-the-art models spanning several families: (i) *traditional CF* (BPRMF [258] and MultiVAE [177]); (ii) *review-based CF* (ConvMF [154] and RMG [340]); (iii) *graph-based CF* (NGCF [325] and LightGCN [126]); (iv) *graph-based CF with attention* (GAT [313] and DGCF [328]).
Reproducibility. We adopt the temporal leave-one-out to split the datasets, where the last two recorded interactions are included in the validation and test. We tune hyper-parameters with [33] and follow the baselines papers, and fix the batch size to 256 and epochs to 400. As for EGCF, we extract review embeddings through a

Table 7.10 Statistics of the tested datasets.

Datasets	**#Users**	**#Items**	**#Interactions**	**Density**	**Average interactions per user**
Baby	4,669	5,435	29,214	0.00115	6.3
Boys & Girls	8,806	4,165	57,928	0.00158	6.6
Men	3,218	7,605	60,299	0.00246	18.7

popular pre-trained model[4]. Datasets and codes are publicly available[5]. All models are implemented in Elliot [11].

Evaluation protocol. We measure the model accuracy by adopting the recall (Recall@k), the normalized discounted cumulative gain (nDCG@k), and the average recall (AR) [126, 328]. Additionally, considering the influence of novel and diverse recommendation lists [310, 311] on both user's and business's interests, we also assess beyond-accuracy metrics such as the expected popularity complement (EPC@k) and the expected free discovery (EFD@k), along with indices measuring concentration and coverage, i.e., the 1's complement of the Gini (Gini@k), the Shannon entropy (SE@k), and the item coverage (iCov@k). Specifically, the EPC@k and the EFD@k refer to long-tail items and stand for the expected number of recommended unknown items which are also relevant, and the expected number of recommended known items which are also relevant, respectively. Furthermore, the Gini@k and the SE@k are used to assess items' distributional inequality, i.e., how unequally a recommender system shows different items to users, and the iCov@k quantifies the number of items that the model recommends. For all metrics, higher values mean better performance.

Results and discussion

Recommendation accuracy. Table 7.11 reports the results for accuracy measures on the top-10 recommendation lists. Surprisingly, the sole introduction of reviews does not seem to produce a consistent accuracy boost. For instance, the strongest review-based method (i.e., RMG) surpasses BPRMF only for the nDCG and the AR on Baby (i.e., 0.0911 vs. 0.0785 and 0.1059 vs. 0.0980, respectively). Contrarily, adopting a graph model can increase the accuracy to traditional CF. When comparing LightGCN with MultiVAE, which obtain the best performance in their respective recommendation families, we observe that the former improves, on Baby, the Recall of 7% and the AR

[4]Please refer to our GitHub repository.

[5]https://github.com/sisinflab/Edge-Graph-Collaborative-Filtering.

of 9%. However, the observed difference even reverts on Men for the nDCG and the AR. The application of attention mechanisms to weight the importance of neighbor nodes is rewarded in Baby and Boys & Girls, where GAT always outperforms NGCF, reaching reARkable results such as the Recall on Baby (i.e., 0.1595 vs. 0.1411) and the AR on Boys & Girls (i.e., 0.1846 vs. 0.1783). Disentangling users' intents on interacted items (i.e., DGCF) produces even more accurate recommendations to NGCF on all datasets. Nevertheless, LightGCN always performs better than DGCF apart from very few cases (i.e., nDCG and AR on Men), even though DGCF's calculated accuracy values do not substantially differ from LightGCN's ones (e.g., see the AR on Baby). Noticeably, the proposed model (i.e., EGCF) outperforms the other baselines under all settings and datasets, with near 100% statistical hypothesis tests (i.e., paired t-test) showing that the results significantly differ. This finding further motivates the goodness of the solution. While we observe a substantial accuracy improvement in traditional and review-based approaches (e.g., +12% to MultiVAE for the AR on Boys & Girls and +53% to RMG for the Recall on Baby), introducing an additional GCN-like network guided by users' reviews is even more beneficial to correct the representation error observable in unweighted graph approaches. Particularly, results show that such correction may lead to small accuracy improvements in some cases (e.g., see the Recall on Boys & Girls when correcting LightGCN) but also larger ones in other cases (e.g., see the nDCG on Men when correcting LightGCN). Such outcomes suggest that *while keeping the error-affected contribution in the final prediction formula is useful to preserve the superior performance of graph-based models to traditional and review-based approaches, the introduced correction term is useful to gain even more accurate preference predictions than unweighted graph architectures.*

Recommendation novelty and diversity. We also assess how novel and diverse recommendation lists are. The two novelty metrics in Table 7.12 (i.e., the EPC@$kandtheEFD$@k, left side) are discussed with concentration and coverage indices (i.e., the Gini@$k, theSE$@k, and the iCov@k, right side) as in an ideal recommender system, a loosely concentrated and large set of recommended items should equally span different ranges of popularity. As previously observed, EGCF is again the best or second-to-best technique. While NGCF is not as capable as LightGCN of proposing long-tail items on Boys & Girls (e.g., 0.2510 vs. 0.3012 for the EFD), the former surpasses the latter for the concentration indices on the same dataset (e.g., 10.5595 vs. 10.1586 for the SE). Since NGCF adopts an ego-neighbor interaction component, the concentration of explored and recommended near items gets loose. Moreover, neighborhood weighting leads to recommend items from the long tail (e.g., comparing

Table 7.11 Accuracy metrics, i.e., Recall, nDCG, and AR, for top-10 lists. Best value is in **bold**, while second-to-best is underlined.

Models	Baby			Boys & Girls			Men		
	Recall	nDCG	AR	Recall	nDCG	AR	Recall	nDCG	AR
MostPop	0.0940	0.0520	0.0627	0.1195	0.0647	0.0776	0.0702	0.0590	0.0672
BPRMF	0.1377	0.0785	0.0980	0.1821	0.1446	0.1666	0.1662	0.1314	0.1527
MultiVAE	0.1768	0.1262	0.1455	0.2224	0.1695	0.1990	0.2091	0.1656	0.1898
ConvMF	0.1230	0.0647	0.0800	0.1146	0.0831	0.0972	0.0838	0.0524	0.0584
RMG	0.1272	0.0911	0.1059	0.1512	0.1065	0.1325	0.1067	0.0727	0.0867
NGCF	0.1411	0.0916	0.1092	0.2006	0.1523	0.1783	0.1969	0.1461	0.1722
LightGCN	0.1892	0.1362	0.1590	0.2305	0.1743	0.2054	0.2124	0.1605	0.1882
GAT	0.1595	0.1051	0.1233	0.2069	0.1573	0.1846	0.1695	0.1254	0.1476
DGCF	0.1874	0.1352	0.1558	0.2249	0.1716	0.2023	0.2070	0.1554	0.1823
EGCF	**0.1944***	**0.1402***	**0.1623***	**0.2325**	**0.1792***	**0.2089***	**0.2195***	**0.1703***	**0.1988***

**statistically significant differences (p-value* ≤ 0.05*).*

GAT with NGCF, we observe a +17% for the EFD on Baby). However, such a finding is not consistent with the trend recognized for the concentration and coverage indices (e.g., when comparing LightGCN with DGCF, we notice 0.1304 vs. 0.2051 for the Gini on Men), as the neighborhood weighting procedure comes at the expense of a limited hop exploration, not allowing such models to explore wider catalog portions. Conversely, injecting user-generated reviews brings new informative content (e.g., RMG recommends a broader and less concentrated range of items from the catalog than DGCF on the Baby dataset). Finally, weighting the neighborhood importance and exploring long-distant user-item interactions through reviews-enriched content (i.e., EGCF) allows to retrieve larger portions of heterogeneous items (e.g., EGCF outperforms LightGCN for the Gini by +63% on Baby and DGCF for the SE by +7% on Boys & Girls), without retaining less popular items from the long-tail (observing the same models, +3% for the EPC on Baby and +6% for the EFD on Boys & Girls). Such outcomes demonstrate that *the content enrichment brought by the extracted review features (injected into the representation error correction) allows to explore user-item interactions at multiple hops, leading to more heterogeneous recommendation lists which also include items from the long-tail.*

Effect of hop exploration number. Figure 7.6 displays, for EGCF, the Recall@k and EFD@k performance variation on top-10 recommendation lists when exploring a number of hops in the range 1-4, where even numbers stand for same node type connections (e.g., user-user), while odd numbers refer to opposite node type connections (i.e., user-item). As evident from the histograms of Baby and Boys & Girls, the Recall@k consistently increases from 1 to 4 hops (this is why we adopt four hop explorations

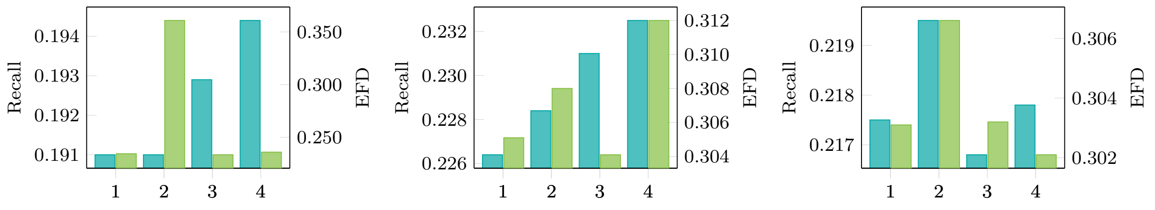
Table 7.12 Calculated novelty metrics, i.e., EPC and EFD, on the left side, and diversity indices, i.e., Gini, SE, and iCov, on the right side, for top-10 lists. Best value is in **bold**, while second-to-best is underlined.

Models	Baby		Boys & Girls		Men	
	EPC	EFD	EPC	EFD	EPC	EFD
MostPop	0.0108	0.0728	0.0135	0.0913	0.0112	0.0904
BPRMF	0.0164	0.1153	0.0306	0.2282	0.0259	0.2167
MultiVAE	0.0268	0.2088	0.0360	0.2874	0.0333	0.2912
ConvMF	0.0135	0.0930	0.0174	0.1219	0.0102	0.0857
RMG	0.0193	0.1488	0.0226	0.1787	0.0144	0.1226
NGCF	0.0194	0.1463	0.0323	0.2510	0.0292	0.2531
LightGCN	0.0289	0.2271	0.0371	0.3012	0.0323	0.2856
GAT	0.0223	0.1708	0.0334	0.2616	0.0248	0.2106
DGCF	0.0287	0.2228	0.0365	0.2945	0.0311	0.2734
EGCF	**0.0298***	**0.2359***	**0.0382***	**0.3120***	**0.0343***	**0.3066***

**statistically significant differences (p-value ≤ 0.05)*

Models	Baby			Boys & Girls			Men		
	Gini	SE	iCov	Gini	SE	iCov	Gini	SE	iCov
MostPop	0.0018	3.5313	18	0.0023	3.5724	18	0.0015	3.9332	32
BPRMF	0.0019	3.7819	40	0.0031	4.0921	203	0.0037	5.2991	192
MultiVAE	0.2139	**9.9160**	4,143	0.2671	10.2463	3,824	0.1085	9.8988	3,014
ConvMF	0.0018	3.5933	18	0.0030	3.9745	220	0.0029	4.6783	265
RMG	0.1059	9.4892	2,130	0.1567	9.7193	2,538	0.1146	10.0344	2,549
NGCF	0.0948	8.8700	2,641	0.3031	**10.5595**	3,668	0.1749	10.7116	3,651
LightGCN	0.1405	9.3105	3,417	0.2398	10.1586	3,647	0.2051	10.8815	4,384
GAT	0.1370	9.2024	3,102	0.2496	10.2821	3,449	0.1235	9.7802	3,530
DGCF	0.0673	8.3193	2,325	0.1800	9.7617	3,208	0.1304	10.2011	3,378
EGCF	**0.2294**	9.8535	**4,490**	**0.3037**	10.4545	**4,030**	**0.2208**	**10.8876**	**4,920**

Statistical significance is not reported since it is calculated only on user level.



(a) Baby (b) Boys & Girls (c) Men

Fig. 7.6 Recommendation performance of EGCF, i.e., *Recall*@k (histogram bars in teal blue) and EFD@k (histogram bars in lime green), on top-10 recommendation lists, when varying the number of explored hops from 1 to 4.

for EGCF on those datasets). The same trend is not observable for Men, where two explored hops seem to provide the highest accuracy boost, motivating the adoption of 2 hop explorations for EGCF on the same dataset. Such behavior could be due to the average number of users' interacted items in Men (approximately 19, see Table 7.10). The node refining probably does not require a broad exploration of its neighborhood. As for the EFD@k, the Baby and the Men datasets seem to agree on two exploration hops to produce the most diverse item lists of recommendations because they leverage (as previously recalled) user-user and item-item interconnections (and similarities). The trend is also aligned with the Boys & Girls dataset, where user-user and item-item links are exploited even at a higher depth (i.e., four exploration hops). The emerged insights shed light on two main contributions: *(i) with the modified neighborhood weighting process, which makes use of reviews to enhance the informative content carried by user-item interactions, EGCF is less limited in the hop exploration, thus providing more accurate recommendations, and (ii) user-user and item-item connections are the keystones on which building more diverse item recommendation lists.*

7.5 Summary

Bringing the lessons-learned from the previous chapters of this thesis, in this chapter, multimodal-aware and graph-based recommender systems were exploited to leverage multimodal information on graph neural networks for recommendation. By exploiting the same experimental and evaluation paradigms already introduced, an initial exploration of the performance of (graph-based) recommender systems using multimodal content was conducted on an extensive set of beyond-accuracy metrics measuring the novelty and diversity of recommendation lists, along with the possible amplification of the negative effects of popularity bias due to the presence of each modality. In this respect, a benchmarking framework for multimodal-aware recommendation was integrated into Elliot to foster the future reproducibility of the performed analysis. In terms of accuracy, the observed results demonstrated how a careful hyper-parameter exploration can lead shallow multimodal approaches (e.g., VBPR) to be competitive to more recent solutions; on the contrary, other recent techniques such as LATTICE show to be consistently outperforming the other baselines (as reported in the related literature). When measuring novelty, diversity, and popularity bias in recommendation, GRCN appeared to be a strong baseline for the diversification of the recommendation lists, but VBPR was the solution reaching the most balanced trade-off. A finer-grained evaluation when separately injecting visual and textual modalities showed how such settings can improve the accuracy but negatively impact the diversity and popularity bias; moreover, a complementary investigation regarding the modalities' influence on metrics pairs outlined that the textual modality has a considerable impact on accuracy but little effect on diversity and popularity bias, whereas the visual modality reduces accuracy while exacerbating popularity bias and limiting the diversity. Conclusively, a new recommendation technique named Edge Graph Collaborative Filtering (ECGF) was designed and implemented. Indeed, by discussing how existing similar solutions are not effectively exploiting (multimodal) features on graph edges in recommendation, we decided to adopt textual features extracted from users' generated reviews as meaningful attributes of graph edges. Specifically, EGCF mitigated the initial issues outlined in the beginning of the thesis; as a positive side effect, EGCF showed the potential to address other algorithmical problems regarding, for instance, over-smoothing in graph neural networks. Results on a number of datasets from the Amazon catalogue, against state-of-the-art review-based recommender systems, confirmed the goodness of the proposed approach on numerous evaluation dimensions considering beyond-accuracy performance and the effect of layer exploration during the message passing.

The current chapter concludes the presentation of results for this thesis. The next chapter will provide key summarizations of the main presented findings from this thesis.

Chapter 8

Conclusion

In the beginning of this thesis work, we outlined two of the most debated open research challenges in personalized recommendation, namely: (i) the inexplicable nature of users' preferences, especially when they are expressed in the form of implicit feedback; (ii) the effective exploitation of the collaborative signal in collaborative filtering-based recommender systems. Recent recommendation approaches leveraging *multimodal deep learning* and *graph neural networks* have shown the potential to address, in a different manner, either the (i) or (ii) issues. Bulding on these two research directions, we decided to provide our contributions to study, formalize, analyze, and address such *macro*-challenges along with their intrinsic *micro*-challenges. Indeed, the previous four chapters widely presented the main outcomes of this thesis, showing the research contributions of the Ph.D. candidate, Daniele Malitesta; the aim of this thesis was to eventually combine the lessons-learned in multimodal-aware and graph-based recommendation to build novel recommendation approaches leveraging the representational power of graph and multimodal learning.

While the outcomes of this thesis were clearly reported in the final "Summary" section of each chapter, in the following, we decide to summarize in a more cohesive manner the key contributions to provide the "bigger picture" behind our work.

Multimodal-aware recommendation. To begin with, Chapter 4 raised the urge to re-formulate the task of multimedia recommendation under the lens of multimodal deep learning. By recognizing the recurrent strategy patterns from the existing literature, the goal was to design a unified formal multimodal schema for multimedia recommendation. The schema was later conceptually validated on specific tasks and scenarios involving multimedia recommendation, opening to further research questions regarding: (i) missing modalities in multimodal recommendation, (ii) pre-trained feature extractors, and (iii) modalities representation.

Then, in Chapter 5, we formally evaluated and provided solutions to the (ii) and (iii) issues, leaving (i) as future work. Specifically, the discussion was tailored to the single scenario of visually-aware recommender systems. Starting from the proposal of a unified framework for the extraction of multimodal features in recommendation, named Ducho, and an extension especially aimed to the reproducibility of visual-based recommender systems, namely, V-Elliot, the two were eventually exploited to benchmark the performance of visually-aware recommender systems with different state-of-the-art convolutional neural network approaches to extract visual features. The evaluation challenges raised from such an analysis were addressed by considering two scenarios and tasks in visually-aware recommendation: fashion recommendation and adversarial attacks/defenses against visually-aware recommender systems. As for the former, a novel approach was proposed with the objective to disentangle the users' preferences at the granularity of content-style properties of fashion items, outperforming other recommendation baselines in the literature on a number of recommendation accuracy and beyond-accuracy metrics. As for the latter, an in-depth study on the effects of adversarially-attacked product images for visual-based recommendation, along with defensive countermeasures, demonstrated how visual attacks may be perceived by the human customer on e-commerce platforms.

Graph-based recommendation. With Chapter 6, we put our focus on the second main topic of this thesis work, namely, graph-based recommender systems. Indeed, the chapter was devoted to a formal introduction and evaluation of strategies and techniques for this family of recommendation algorithms. Similarly to what was done for visually-aware recommendation, the chapter begun with the proposal of a framework, built as an extension of Elliot, for the reproducibile and rigorous evaluation of state-of-the-art graph-based recommender systems. The framework was used as a useful tool to perform a comprehensive reproducibility analysis on recommendation systems using graph neural networks, by underlining interesting findings regarding the possible influence of dataset characteristics (in the form of node degree) on the performance of such models. The analysis was extended to topological properties of the user-item bipartite and undirected graphs; this confirmed the strong influence of such dataset characteristics on recommendation performance, and unveiled how the latent-factor strategy may be the dominating component of any graph-based recommender system, providing a novel perspective on graph collaborative filtering. Finally, by recognizing node representation and neighborhood exploration as the core strategies underlying any graph-based recommender system, we decided to explore them on accuracy and beyond-accuracy metrics, in single- and multi-objective experimental settings. Results underlined that user-user and item-item message-passing explorations may be beneficial to meet accuracy/beyond-accuracy trade-offs, while we also noticed how implicit message-passing in recent approaches could be harmful to the consumer-provider fairness scenario.

Graph-based recommendation leveraging multimodal information. Finally, in Chapter 7, multimodal-aware and graph-based recommendation were eventually combined into a unique recommendation framework. First, an investigating analysis of the accuracy/beyond-accuracy performance of graph-based approaches leveraging multimodal information was conducted through an extensive benchmark in Elliot. Results demonstrated that, differenlty from what stated in the literature, careful hyper-parameter explorations can bring some shallow multimodal approaches (e.g., VBPR) to be competitive with recent solutions; moreover, finer-grained evaluations proved the negative effects on performance of the considered modalities when injected separately. On such bases, a novel approach named Edge Graph Collaborative Filtering (ECGF) was proposed. Through the exploitation of textual features of users' generated reviews on graph edges (something that is rarely seen in the related literature) the model was capable of outperforming existing solutions on accuracy and beyond-accuracy metrics, while tackling other algorithmic problems regarding, for instance, over-smoothing in graph neural networks.

Towards future work. As a final statement, we believe the proposed thesis represents just the beginning of a longer and increasingly-interesting research path. Despite the numerous conducted analyses and proposals presented in this work, other research questions and open challenges naturally aroused. That is why we intentionally decided not to report the possible future work of the thesis in this chapter. Indeed, several initial research directions and ideas have been formalized and tested over the last few months before the submission of the thesis; their preliminary outcomes will be presented in the next (and last) chapter, following the same thematic structure provided in this chapter. As a disclaimer, we want to state that it is likely most of these ideas will not be working when further explored and investigated. However, if even the 1% of those should provide at least interesting insights, that would be consider as a true success.

Chapter 9

Future directions

This last chapter outlines ideas and/or initial experimental analyses that have been conducted over the last few months before (and during) the writing of the thesis. To better organize the narrative, we decide to group the following paragraphs into three main sections, which are aligned with the three main research paths of this thesis: (i) multimodal-aware recommendation, (ii) graph-based recommendation, and (iii) graph-based recommendation leveraging multimodal information.

9.1 Multimodal-aware recommendation

9.1.1 Domain-specific multimodal features

Given the limitations imposed by the adoption of pre-trained multimodal features (see again Section 4.5.2), we wish to underline the benefits of *domain-specific* features in the multimodal schema we have outlined. Extracting such high-level features from input data would entail injecting meaningful and task-aware informative content into the recommendation system, thereby better-profiling items and users on the platform to generate more tailored recommendations. Domain-specific features should necessitate domain-specific extraction models, which may have been previously trained and optimized on similar tasks to the one we are pursuing. Regarding *fashion* recommendation, for instance, we recall the work by [105], a pre-trained architecture for the comprehensive visual analysis of clothing photos. Another example is the *food* recommendation system proposed by [357], which analyzes food-related photos.

Furthermore, in the field of *audio* and *text* understanding and classification, Choi et al. [73] construct a deep model based on convolutional recurrent neural networks for music tagging by taking into account songs' local features and temporal characteristics, whereas the work in [30] tackles sentiment analysis in user-generated tweets.

9.1.2 Multimodality on user-item interactions

Multimodality is the most intuitive approach to describe the nature of items in multimedia recommendation [86, 87], but this does not hold for the users' profile.

First, from a technical point of view, profiling each user through multimodal features (e.g., her voice, her visual appearance) would require sophisticated technologies that users' digital devices could not necessarily support (e.g., smartphones). Second, from a practical point of view, it is likely that users would not be disposed to share such personal data on online platforms, primarily for privacy concerns. Despite the raised critical aspects, a few examples from the literature [305, 339] propose to model the user profile in such a way that her preferences toward each multimodal aspect of items are made explicit and learned during model's training. However, these systems rely solely on the multimodal profiles of the items, disregarding alternative information sources. *Product reviews*, which express opinions and comments about items that have been clicked, watched, or purchased, could be a valuable tool for revealing users' nuanced preferences toward each item in the recommendation system.

Existing review-based approaches [323, 399] work by integrating reviews as the *textual modality* to represent *items*. However, we believe that a more logical and effective way to integrate reviews would be to view them as a medium to represent *user* preference *over items*, thereby providing additional and complementary preference scores in addition to numeric ratings or implicit feedback that are typically used to compute recommendations. Such reasoning may be easily generalized to include user-generated data regarding interacting things (such as images or videos of delivered products), which we can characterize as *multimodal feedback* (see Figure 9.1). When compared to numerical feedback, which tends to be *atomic* (single-faceted), multimodal feedback could be considered as *composite*, revealing nuance and the user's multi-faceted opinion of the products [19].

9.1.3 Fine-grained multimodal features

Multimodality is a way to effectively profile the multi-faceted aspects of items and users' preference (e.g., I bought this smartphone because its technical *description* is quite exhaustive and its *display* amazes me; I like this song since I love the *music* and the *lyrics*). Nevertheless, analyzing and learning users' tastes at modalities' granularity

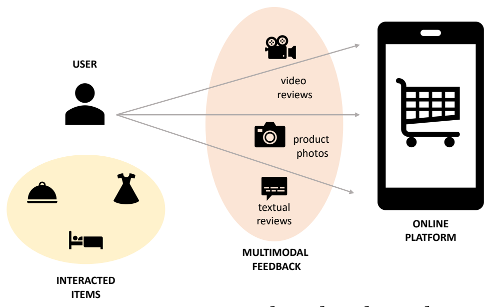


Fig. 9.1 An example of how users generate and upload *multimodal feedback* about interacted items (e.g., textual reviews, product photos, or even video reviews) on online platforms. Such *user-item* sources of information may be suitably exploited to better profile user' preferences.

might not be enough to uncover all aspects underlying every user-item interaction. In contexts where modalities bring a great source of heterogeneous information, a finer-grained feature processing could help better unveil hidden facets. For instance, when it comes to the recommendation of fashion items (e.g., dresses, shoes, jewelry), user attention may be captivated by specific item visual characteristics, such as colors, shapes, and particular patterns and motifs [83]. Similarly, a song involves several features [178] (i.e., pitch, rhythm, and dynamics), which could differently influence users' attitudes towards it. Uncovering and understanding details at this finer granularity should be one of the main directions toward the novel recommendation approaches of multimedia products and services.

9.1.4 An extensive evaluation of multimedia recommender systems

To date, very limited effort has been put into the extensive evaluation of multimedia recommender systems. The principal reason is that, apart from some recent frameworks [269, 404] which integrate multimedia recommender systems into their pipelines, each novel multimedia recommender system introduces its own implementation of the proposed approach with different dataset pre-processing solutions, sampling strategies, and evaluation protocols. Indeed, this may undermine the fair comparison of multimedia recommender systems, which cannot benefit from shared and unified training and evaluation frameworks to run rigorous and reproducible experiments as in other recommendation domains and scenarios [11, 397]. To this end, we plan to start from the initial benchmarking analysis we proposed in this work to further assess the reproducibility of the tested baselines. On such basis, the next steps would be to evaluate the recommendation performance under more comprehensive experimental settings involving, for instance, (i) a larger plethora of pre-trained deep learning models for the extraction of multimodal features; (ii) other multimodal datasets involving all modalities (as our framework offers the possibility to inject visual, textual, and audio features); (iii) a more careful evaluation of such models under beyond-accuracy recommendation metrics [206, 207].

9.2 Graph-based recommendation

9.2.1 Topological properties in graph collaborative filtering

With reference to the work presented in [209], we plan to extend the proposed analysis to assess the impact of topological dataset characteristics on other recommendation metrics accounting for the novelty and diversity of the produced recommendation lists, and potential biases and fairness issues in recommendation; this investigation may be conducted by utilizing ad-hoc graph recommender systems specifically designed to optimize such objectives. Furthermore, we intend to directly create the synthetic user-item graphs from scratch with graph generator techniques which may resemble real-world recommendation data but with desired topological properties. Finally, we seek to investigate the impact of other topological aspects of the user-item graph on the performance of graph-based recommender systems, such as the presence of user, item, and user-item communities and their inter dependencies, as this direction is poorly explored in the related literature so far.

9.2.2 Bridging recommendation and link prediction

Item recommendation (the task of predicting if a user may interact with new items from the catalogue in a recommendation system) and link prediction (the task of identifying missing links in a knowledge graph) have long been regarded as distinct problems. As future research direction, we seek to show that the item recommendation problem can be seen as an instance of the link prediction problem, where entities in the graph represent users and items, and the task consists in predicting missing instances of the relation type interactsWith. To this aim, we start by systematically benchmarking some factorisation-based link prediction models on recommendation tasks against graph-based recommendation approaches, showing that predictive accuracy of the former is better than or competitive with most of the selected recommendation models.

Before providing the initial experimental results we obtained so far, we briefly introduce the formalization for popular factorization-based link prediction approaches. Then, we select three of them in our analysis, namely, DistMult [354], CP [132], and

ComplEx [308]. Note that this preliminary work has been conducted (and is still going further) during Daniele Malitesta's internship at the University of Edinburgh under the supervision of Dr. Pasquale Minervini.

Preliminaries

A Knowledge Graph $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ contains a set of subject-predicate-object $\langle s,p,o \rangle$ triples, where each triple represents a relationship of type $p \in \mathcal{R}$ between the subject $s \in \mathcal{E}$ and the object $o \in \mathcal{E}$ of the triple. Here, $\mathcal{E}$ and $\mathcal{R}$ denote the set of all entities and relation types, respectively. However, many real-world knowledge graphs are largely incomplete [89, 90, 97, 229] – link prediction focuses of the problem of identifying missing links in (possibly very large) knowledge graphs.

More formally, given an incomplete graph $\mathcal{G}^- \subset \mathcal{G}$, where $\mathcal{G}$ denotes a complete graph, the task consists of identifying the triples $\langle s,p,o \rangle$ triples such that $\langle s,p,o \rangle \notin \mathcal{G}^-$ and $\langle s,p,o \rangle \in \mathcal{G}$.

Neural link predictors

A *neural link predictor* differentiable model where entities in $\mathcal{E}$ and relation types in $\mathcal{R}$ are represented in a continuous embedding space, and the likelihood of a link between two entities is a function of their representations.

More formally, neural link predictors are defined by a parametric *scoring function* $\phi_\theta : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \mapsto \mathbb{R}$, with parameters θ that, given a triple $\langle s,p,o \rangle$, produces the likelihood that entities s and o are related by the relationship p.

Scoring functions

Neural link prediction models can be characterised by their scoring function ϕ_θ. For example, in TransE [40], the score of a triple $\langle s,p,o \rangle$ is given by

$$\phi_\theta^{\text{TransE}}(s,p,o) = - \left\| \mathbf{s} + \mathbf{p} - \mathbf{o} \right\|_2, \tag{9.1}$$

where $\mathbf{s}, \mathbf{p}, \mathbf{o} \in \mathbb{R}^k$ denote the embedding representations of s, p, and o, respectively. In DistMult [354], the scoring function is defined as follows:

$$\phi_\theta^{\text{DistMult}}(s,p,o) = \langle \mathbf{s}, \mathbf{p}, \mathbf{o} \rangle = \sum_{i=1}^{k} \mathbf{s}_i \mathbf{p}_i \mathbf{o}_i, \tag{9.2}$$

where $\langle \cdot, \cdot, \cdot \rangle$ denotes the tri-linear dot product. Canonical Tensor Decomposition [CP, 132] is similar to DistMult, with the difference that each entity x has two representations, $\mathbf{x}_s \in \mathbb{R}^k$ and $\mathbf{x}_o \in \mathbb{R}^k$, depending on whether it is being used as a subject or object:

$$\phi_\theta^{\text{CP}}(s,p,o) = \langle \mathbf{s}_s, \mathbf{p}, \mathbf{o}_o \rangle. \tag{9.3}$$

In RESCAL [230], the scoring function is a bilinear model given by:

$$\phi_\theta^{\text{RESCAL}}(s,p,o) = \mathbf{s}^\top \mathbf{P} \mathbf{o}, \tag{9.4}$$

where $\mathbf{s}, \mathbf{o} \in \mathbb{R}^k$ is the embedding representation of s and p, and $\mathbf{P} \in \mathbb{R}^{k \times k}$ is the representation of p. Note that DistMult is equivalent to RESCAL if $\mathbf{P}$ is constrained to be diagonal. Another variation of this model is ComplEx [308], where the embedding representations of s, p, and o are complex vectors – i.e. $\mathbf{s}, \mathbf{p}, \mathbf{o} \in \mathbb{C}^k$ – and the scoring function is given by:

$$\phi_\theta^{\text{ComplEx}}(s,p,o) = \Re(\langle \mathbf{s}, \mathbf{p}, \overline{\mathbf{o}} \rangle), \tag{9.5}$$

where $\Re(\mathbf{x})$ represents the real part of $\mathbf{x}$, and $\overline{\mathbf{x}}$ denotes the complex conjugate of $\mathbf{x}$. In TuckER [27], the scoring function is defined as follows:

$$\phi_\theta^{\text{TuckER}}(s,p,o) = \mathbf{W} \times_1 \mathbf{s} \times_2 \mathbf{p} \times_3 \mathbf{o}, \tag{9.6}$$

where $\mathbf{W} \in \mathbb{R}^{k_s \times k_p \times k_o}$ is a three-way tensor of parameters, and $\mathbf{s} \in \mathbb{R}^{k_s}$, $\mathbf{p} \in \mathbb{R}^{k_p}$, and $\mathbf{o} \in \mathbb{R}^{k_o}$ are the embedding representations of s, p, and o. For the moment, we mainly focus on DistMult, CP, and ComplEx due to their effectiveness on several link prediction benchmarks [140, 265].

Training objectives

Another dimension for characterising neural link predictors is their *training objective*. Early neural link prediction models such as RESCAL and CP were trained to minimise the reconstruction error of the whole adjacency tensor [161, 231, 314].

To scale to larger Knowledge Graphs, subsequent approaches such as Bordes et al. [40] and Yang et al. [354] simplified the training objective by using *negative sampling*: for each training triple, a corruption process generates a batch of negative examples by corrupting the subject and object of the triple, and the model is trained by increasing the score of the training triple while decreasing the score of its corruptions. More formally, the loss is given by $\mathcal{L}(\mathcal{G}) = \sum_{\langle s,p,o \rangle} \ell(s,p,o)$ with:

$$\ell(s,p,o) = \sum_{\langle \hat{s},p,\hat{o}\rangle \in \mathcal{N}(s,p,o)} [\gamma - \phi(s,p,o) + \phi(\hat{s},p,\hat{o})]_+ , \tag{9.7}$$

where $\mathcal{N}(s,p,o) = \{\langle \hat{s},p,o\rangle \mid s \neq \hat{s}\} \cup \{\langle s,p,\hat{o}\rangle \mid o \neq \hat{o}\}$ denotes the set of triples obtained by corrupting the training triple $\langle s,p,o\rangle$.

This approach was later extended by Dettmers et al. [90] where, given a subject s and a predicate p, the task of predicting the correct objects is cast as a $|\mathcal{E}|$-dimensional multi-label classification task, where each label corresponds to a distinct object and multiple labels can be assigned to the (s,p) pair. This training objective is referred to as KvsAll by Ruffinelli et al. [265] and can be formalised as $\mathcal{L}(\mathcal{G}) = \sum_{e_1 \in \mathcal{E}} \sum_{p \in \mathcal{R}} \ell(e_1,p)$, with:

$$\ell(e_1,p) = \sum_{e_2 \in \mathcal{E}} \mathrm{BCE}(e_2,p,e_1) + \mathrm{BCE}(e_1,p,e_2), \tag{9.8}$$

with $\mathrm{BCE}(s,p,o) = -(y \log p + (1-y)\log(1-p))$, $p = \sigma(\phi(s,p,o))$, and $y = \mathbb{1}[\langle s,p,o\rangle \in \mathcal{G}]$.

Another extension was proposed by Lacroix et al. [165] where, given a subject s and an object p, the task of predicting the correct object o in the training triple is cast as a $|\mathcal{E}|$-dimensional multi-class classification task, where each class corresponds to a distinct object and only one class can be assigned to the (s,p) pair; this is referred to as 1vsAll by Ruffinelli et al. [265], and defined as $\mathcal{L}(\mathcal{G}) = \sum_{\langle s,p,o\rangle} \ell_{\mathrm{s}}(s,p,o) + \ell_{\mathrm{o}}(s,p,o)$, with:

$$\begin{aligned} \ell_{\mathrm{s}}(s,p,o) &= -\phi(s,p,o) + \log\left[\sum_{\hat{s}} \exp\left(\phi(\hat{s},p,o)\right)\right], \\ \ell_{\mathrm{o}}(s,p,o) &= -\phi(s,p,o) + \log\left[\sum_{\hat{o}} \exp\left(\phi(s,p,\hat{o})\right)\right]. \end{aligned} \tag{9.9}$$

Regularisers

As noted by Bordes et al. [40], imposing regularisation terms on the learned entity and relation representations prevents the training process from trivially optimising the training objective by increasing the embedding norms. Early works such as Bordes et al. [40, 41], Glorot et al. [112], and Jenatton et al. [143] proposed constraining the embedding norms. More recently, Trouillon et al. [308] and Yang et al. [354] proposed adding a L_2 regularisation term on entity and relation representations to the training objective. Lastly, Lacroix et al. [165] observed systematic improvements by replacing the L_2 norm with a nuclear tensor 3-norm.

Item recommendation as link prediction

Note that item recommendation models can be cast as a particular case of link prediction in Knowledge Graphs.

More specifically, user-item score predictors $\rho_\theta : \mathcal{U} \times \mathcal{I} \mapsto \mathbb{R}$ can be seen as learning a ranking between missing triples in a Knowledge Graph $\mathcal{G}$, where the set of entities corresponds to the union of the sets of users and items $\mathcal{E} = \mathcal{U} \cup \mathcal{I}$, the set of relations corresponds to a single relation $\mathcal{R} = \{\text{interactsWith}\}$, and the graph $\mathcal{G}$ to complete is given by the observable interactions between users and items:

$$\mathcal{G} = \{\langle u, \text{interactsWith}, i\rangle \mid x_{ui} = 1\}. \tag{9.10}$$

This enables the off-the-shelf application of state-of-the-art neural link prediction methods to the item recommendation problem.

Initial experimental results

Initial experiments were conducted to test the efficacy of state-of-the-art link prediction approaches (i.e., CP, DistMult, and ComplEx) when trained and tested for the task of item recommendation, on the Yelp-2018 dataset already used for other publications presented in this thesis [20, 210]. In this respect, the link prediction models were compared against popular graph-based recommendation systems, whose results have been directly picked from the above publications as the experimental settings were exactly the same.

While graph-based recommender systems were tuned and evaluated through the framework Elliot, the link prediction models were tuned and evaluated with the popular framework LibKGE for knowledge graph embeddings [264]. However, as the latter is designed to address the task of link prediction which, as stated, is a generalization of the recommendation task, modifications were needed to make the link prediction results comparable to those of item recommendation. Apart from a re-casting of the recommendation dataset into a knowledge graph one (users and items are entities connected through one single interaction type, interactsWith), the calculation of the Recall@k metric was added to LibKGE, as its formulation in recommender systems is different from those utilized in link prediction-alike tasks. As for the nDCG, LibKGE was modified in a way the recommendation lists for each user were stored after the training of each model. Thus, Elliot was eventually exploited to calculate the nDCG@k.

In terms of evaluation paradigm, we used the exact same train/test splitting adopted for the training and test of the graph-based recommender systems. However, in order

Table 9.1 Results of state-of-the-art item recommendation and link prediction models trained and evaluated for the task of recommendation. The metrics are Recall@20 and nDCG@20, and the selected dataset is Yelp-2018.

Performance	**Item Recommendation**								**Link Prediction**		
	UserkNN	ItemkNN	NGCF	DGCF	LightGCN	SGL	UltraGCN	GFCF	CP	DistMult	ComplEx
Recall	0.0630(**5**)	0.0610	0.0556	0.0621	0.0629	0.0669(**3**)	0.0672(**2**)	0.0697(**1**)	0.0367	0.0465	0.0662(**4**)
nDCG	0.0528(**5**)	0.0507	0.0452	0.0505	0.0516	0.0552(**3**)	0.0553(**2**)	0.0571(**1**)	0.0293	0.0375	0.0539(**4**)

to perform hyperparameter tuning of the link prediction approaches, we retained the 10% of the training set as validation set, and used the Recall@20 as validation metric. Three different validation splittings were involved to provide an as much as possible generalization of the validation set. Regarding the explored hyperparameters, and similarly to what done in [264], we tested three training strategies (i.e., 1vsAll, KvsAll, and negative sampling), two losses (i.e., BCE and KL), two batch sizes (i.e., 1024 and 2048), two optimizers (i.e., Adam and Adagrad), four learning rates (i.e., 0.0001, 0.001, 0.01, 0.1), two regularizers (i.e., N3, LP), and six regularization weights (i.e., 1e-06, 1e-05, 1e-04, 1e-03, 1e-02, 1e-01). The embedding size was kept fixed at 64 to be consistent with the settings followed for the graph-based recommender systems.

On Yelp-2018, the results (i.e., Table 9.1) show that link prediction approaches are quite competitive with respect to graph-based recommender systems on the task of recommendation, especially ComplEx. Indeed, the latter model is always in the top-5 performing models in the results, outperforming powerful and recent approaches in the recommendation literature, such as NGCF, DGCF, and LightGCN. As already stated, the results are quite preliminary, and more careful experiments and analyses should be taken into consideration. At the moment, we are running additional experiments on other recommendation datasets and other recommendation models, trying to explore larger hyperparameter spaces to uncover possible relations between specific parameter values and improved results provided by the link prediction models (as done in [264]).

9.2.3 How powerful is adjacency normalization for recommendation?

As outlined in the background section regarding graph neural networks (Chapter 3), the normalization of the adjacency matrix has been proven to be quite beneficial for the training of graph neural networks, also when it comes to graph-based recommendation:

$$\mathbf{A}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}. \tag{9.11}$$

However, driven by the observed dependencies between topological properties (especially degree assortativity) and recommendation performance, one natural question arises: "how useful and powerful is adjacency normalization in recommendation?".

The question makes even more sense if considering the frequency of co-occurrences of degree pairs in the user-item interaction graph. Specifically, if we re-formulate the symmetric adjacency normalization at node-level instead of graph-level, we have (e.g., in the case of LightGCN):

$$\mathbf{e}_u^{(l+1)} = \sum_{i \in \mathcal{N}_u} \frac{\mathbf{e}_i^{(l)}}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}, \tag{9.12}$$

where we have the co-occurrence of degree $|\mathcal{N}_u|$ and $|\mathcal{N}_i|$ in the same message-passing formulation for nodes u and i.

By observing the frequency of co-occurrencies of node degrees along with the contour plot for the symmetric adjacency normalization on two recommendation datasets (i.e., Allrecipes and Gowalla), it becomes evident how the latter does not evidently vary in the trend, while the former is highly dataset-dependent. This preliminary intuition may pave the way to a re-formulation of the adjacency normalization, which could be aware of advances topological properties of the dataset it is applied on.

9.3 Graph-based recommendation leveraging multimodal information

9.3.1 Exploiting reviews on user-user and item-item graphs

Recent works [382] propose to use multimodal content data (e.g., product images and descriptions) to build the item-item similarity graph, which has been shown to enhance the information carried by the user-item interaction graph. Exploiting the sole multimodal profile of items could fail to measure their similarities from the **collaborative filtering** perspective. There exist other sources of information (e.g., users' generated reviews) that can suitably model items similarity according to users' actual opinions about them. We propose to exploit high-level features extracted from reviews to infer **item-item**, but also **user-user** similarities, in the collaborative filtering spirit: (i) users interacting with the same items may share similar preferences; (ii) items viewed by the same users may be similar from users' perspectives.

The proposed framework

The proposed framework mainly builds on the preprocessing of review features to build the user-user (U-U) and item-item (I-I) graphs. Once the two graphs have been initialized, they can be integrated seamlessly into any recommendation backbone based upon the learning of user and item embeddings.

Building the U-U and I-I graphs: basic intuitions

We report the basic intuitions about the U-U graph (the same applies to the I-I graph). User u_1 and user u_2 might share similar preferences if: (BI_1) their interacted item sets have elements in common (i.e., co-occurrences); (BI_2) the polarity of the opinions they expressed about them is aligned. In this respect, we observe some problems and/or limitations in the current intuitions, namely: is it safe to infer that two users are similar if they only co-interacted with very few items? What does it mean to state that the polarity of the opinions expressed is aligned?

Building the U-U and I-I graphs: advanced intuitions

We come up with more advanced intuitions. Two users u_1 and u_2 might share similar preferences if: (AI_1) their clustering coefficient (intersection over union/max/min) is close to 1; (AI_2) the number of co-interacted items over the average number of co-interacted items is close to 1.

Introducing review features

Previous works [62, 286] have shown that reviews can convey the multi-faceted opinions that user have on products they interacted with. This is more than a simple rating value. For this reason, we: (i) extract high-level features from reviews through pre-trained sentiment analysis deep neural networks; (ii) measure the similarity (cosine similarity) among reviews that u_1 and u_2 have written about the same items; (iii) sum such similarities across all co-interacted items for u_1 and u_2; (iv) weight such a value according to the clustering coefficient (intersection over union/max/min), or the number of interacted items over the average number of co-interacted items.

Ablation study and sparsification

To actually demonstrate the rationale of our proposed approach, we also decide to test the framework on other weighting settings: no weight with sum of cosine similarities among review features; same weighting procedures as above but with ratings (encoded through binary notation) instead of review features; no weight with sum of cosine similarities among ratings (encoded through binary notation) instead of review features. As observed by previous works [382], the obtained user-user similarity matrix could be too much dense. We apply *knn*-based sparsification (top-k is set from 10 to 100 with 10 as step).

Framework training

Once the U-U and I-I similarity graphs are built, we freeze them during the framework training. Given a backbone (e.g., matrix factorization), we perform the following step (for each batch): (i) propagate user and item ***similarity*** embeddings on the U-U and I-I graphs (LightGCN layer for the message passing); (ii) aggregate such embeddings on the usual ***collaborative*** ones through a weighted sum (alpha and beta coefficients as hyper-parameters to perform such a weighted sum on user and item, respectively); (iii) predict the user-item score on the batch according to the backbone.

9.3.2 A feature propagation approach for missing modalities

As already discussed in the open challenges from Chapter 4, the unavailability of modalities in multimodal-aware recommendation represents one of the most debated issues in the current literature. However, to the best of our knowledge, very few and domain-specific solutions [317] have been proposed in the literature so far.

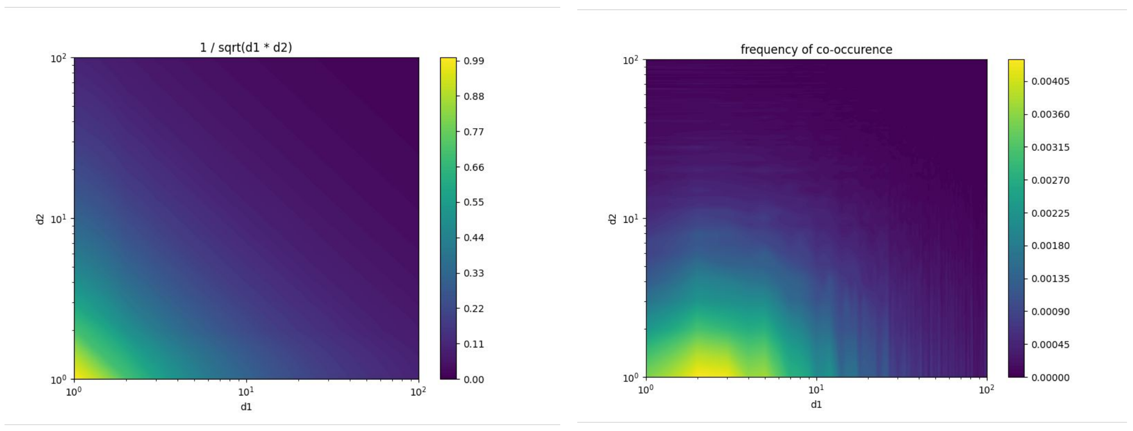
Given that most of the recent approaches in multimodal-aware recommendation are built upon graph neural network architectures, we take into account the recent work by Rossi et al. [263]. In the paper, the authors propose a simple but effective approach to deal with missing node features in graph representation learning by proposing a feature propagation approach as preliminary phase before the actual training of the model for the downstream task. Let us suppose to be addressing a classical task involving graph neural networks, for instance, node classification; then, let us assume that a certain percentage of the node features are missing from the original graphs. The authors' intuition is to run a preliminary multi-step feature propagation procedure to approximate the missing node features through:

$$\mathbf{E}^{(l+1)} = \mathbf{A}\mathbf{E}^{(l)}, \tag{9.13}$$

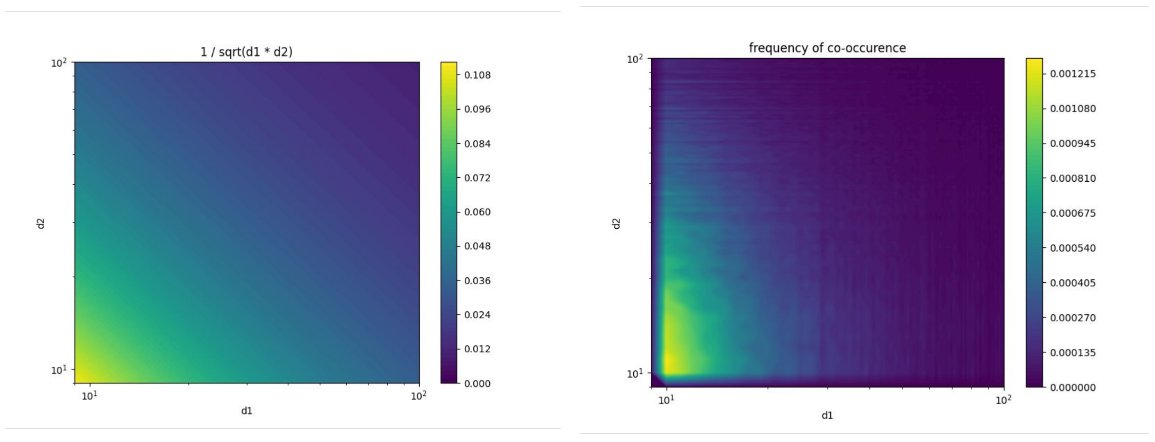
where, for the sake of simplicity, we indicate the features for all nodes as $\mathbf{E}$, and the adjacency matrix as $\mathbf{A}$. Repeating the feature propagation for a certain number of steps allows to obtain an approximated representation for the missing features (the existing ones are not modified after the complete feature propagation), reaching comparable if superior performance to the ones obtained by other similar existing strategies.

Inspired by the same strategy, we decide to apply feature propagation in the recommendation setting where some multimodal features of items may be missing. Limiting the analysis to a two-modalities setting (e.g., visual and textual) two cases may occur: (i) a percentage p of features is missing from both the modalities, (ii) a percentage p_1 and a percentage p_2 are missing from the visual and textual modalities. Note that, to apply the feature propagation approach, we should first cast the solution proposed in [263] to the recommendation scenario, which involves a bipartite and undirected graph. As the feature propagation approach has been specifically tailored to monopartite graphs, we first propose to obtain the projected item-item graph from the user-item one. Then, feature propagation may be applied for a certain number of iterative steps.

We performed some preliminary experiments by considering the model proposed in [336] (called MMSSL) on the Amazon Baby dataset. For simplicity, we limit our investigation to the case in which an equal percentage of missing features is missing for both modalities. Note that having a $p\%$ of missing modalities on both modalities means to have $p\%$ random items from the catalogue whose visual and textual modalities are missing. Results against shallow feature replacement approaches (e.g., put all missing features to zero, consider the mean of the existing features, and put the missing feature at random) show some interesting insights (see Figure 9.3). Indeed, the feature propagation approach seems to perform slightly better (in some cases much better) than the other baselines, paving the way to more nuanced analyses and experiments.



(a) Allrecipes



(b) Gowalla

Fig. 9.2 Contour plot of the symmetric adjacency normalization (left-side in each subfigure) alongside the node degree co-occurrence (right-side in each subfigure) for (a) Allrecipes and (b) Gowalla datasets.

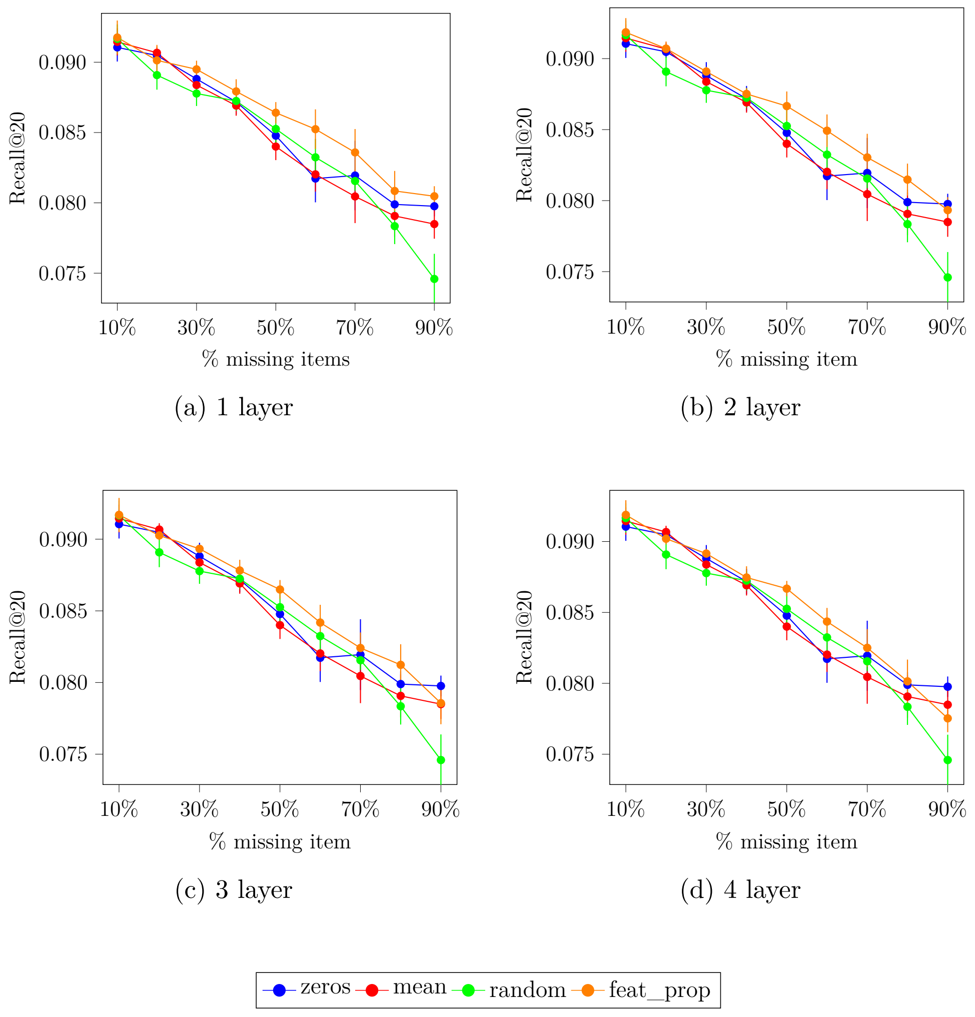


Fig. 9.3 Feature propagation applied for varying percentages of items with missing modalities for the MMSSL model trained on Amazon Baby.

Bibliography

[1] Himan Abdollahpouri. “Popularity Bias in Ranking and Recommendation.” In: *AIES*. ACM, 2019, pp. 529–530.

[2] Himan Abdollahpouri, Gediminas Adomavicius, Robin Burke, Ido Guy, Dietmar Jannach, Toshihiro Kamishima, Jan Krasnodebski, and Luiz Augusto Pizzato. “Multistakeholder recommendation: Survey and research directions.” In: *User Model. User Adapt. Interact.* 30.1 (2020), pp. 127–158.

[3] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. “Controlling Popularity Bias in Learning-to-Rank Recommendation.” In: *RecSys*. ACM, 2017, pp. 42–46.

[4] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. “Managing Popularity Bias in Recommender Systems with Personalized Re-Ranking.” In: *FLAIRS*. AAAI Press, 2019, pp. 413–418.

[5] Himan Abdollahpouri, Mehdi Elahi, Masoud Mansoury, Shaghayegh Sahebi, Zahra Nazari, Allison Chaney, and Babak Loni. “MORS 2021: 1st Workshop on Multi-Objective Recommender Systems.” In: *RecSys*. ACM, 2021, pp. 787–788.

[6] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. “MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing.” In: *ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 21–29.

[7] Gediminas Adomavicius and Alexander Tuzhilin. “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions.” In: *IEEE Trans. Knowl. Data Eng.* 17.6 (2005), pp. 734–749.

[8] Gediminas Adomavicius and Jingjing Zhang. “Impact of data characteristics on recommender systems performance.” In: *ACM Trans. Manag. Inf. Syst.* 3.1 (2012), 3:1–3:17.

[9] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron C. Courville. “Learning Distributed Representations from Reviews for Collaborative Filtering.” In: *RecSys*. ACM, 2015, pp. 147–154.

[10] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco M. Donini, Eugenio Di Sciascio, and Tommaso Di Noia. “The Challenging Reproducibility Task in Recommender Systems Research between Traditional and Deep Learning Models.” In: *SEBD*. Vol. 3194. CEUR Workshop Proceedings. CEUR-WS.org, 2022, pp. 514–521.

[11] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. "Elliot: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation." In: *SIGIR*. ACM, 2021, pp. 2405–2414.

[12] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. "V-Elliot: Design, Evaluate and Tune Visual Recommender Systems." In: *RecSys*. ACM, 2021, pp. 768–771.

[13] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, Eugenio Di Sciascio, and Tommaso Di Noia. "How to Perform Reproducible Experiments in the ELLIOT Recommendation Framework: Data Processing, Model Selection, and Performance Evaluation." In: *IIR*. Vol. 2947. CEUR Workshop Proceedings. CEUR-WS.org, 2021.

[14] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Dietmar Jannach, and Claudio Pomo. "Top-N Recommendation Algorithms: A Quest for the State-of-the-Art." In: *UMAP*. ACM, 2022, pp. 121–131.

[15] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, and Claudio Pomo. "Reenvisioning the comparison between Neural Collaborative Filtering and Matrix Factorization." In: *RecSys*. ACM, 2021, pp. 521–529.

[16] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. "A Study of Defensive Methods to Protect Visual Recommendation Against Adversarial Manipulation of Images." In: *SIGIR*. ACM, 2021, pp. 1094–1103.

[17] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Daniele Malitesta, Vincenzo Paparella, and Claudio Pomo. "Auditing Consumer- and Producer-Fairness in Graph Collaborative Filtering." In: *ECIR (1)*. Vol. 13980. Lecture Notes in Computer Science. Springer, 2023, pp. 33–48.

[18] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, Daniele Malitesta, and Claudio Pomo. "How Neighborhood Exploration influences Novelty and Diversity in Graph Collaborative Filtering." In: *MORS@RecSys*. Vol. 3268. CEUR Workshop Proceedings. CEUR-WS.org, 2022.

[19] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, Daniele Malitesta, and Claudio Pomo. "Reshaping Graph Recommendation with Edge Graph Collaborative Filtering and Customer Reviews." In: *DL4SR@CIKM*. Vol. 3317. CEUR Workshop Proceedings. CEUR-WS.org, 2022.

[20] Vito Walter Anelli, Daniele Malitesta, Claudio Pomo, Alejandro Bellogín, Eugenio Di Sciascio, and Tommaso Di Noia. "Challenging the Myth of Graph Collaborative Filtering: a Reasoned and Reproducibility-driven Analysis." In: *RecSys*. ACM, 2023, pp. 350–361.

[21] Vito Walter Anelli, Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. “Assessing Perceptual and Recommendation Mutation of Adversarially-Poisoned Visual Recommenders (short paper).” In: *DP@AI*IA*. Vol. 2776. CEUR Workshop Proceedings. CEUR-WS.org, 2020, pp. 49–56.

[22] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Daniele Malitesta, and Felice Antonio Merra. “Adversarial Attacks against Visual Recommendation: an Investigation on the Influence of Items’ Popularity.” In: *OHARS@RecSys*. Vol. 3012. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 33–44.

[23] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzurra Ragone, and Joseph Trotta. “How to Make Latent Factors Interpretable by Feeding Factorization Machines with Knowledge Graphs.” In: *ISWC (1)*. Vol. 11778. Lecture Notes in Computer Science. Springer, 2019, pp. 38–56.

[24] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. “A Simple but Tough-to-Beat Baseline for Sentence Embeddings.” In: *ICLR (Poster)*. OpenReview.net, 2017.

[25] Nabiha Asghar. “Yelp Dataset Challenge: Review Rating Prediction.” In: *CoRR* abs/1605.05362 (2016).

[26] Ricardo Baeza-Yates. “Bias in Search and Recommender Systems.” In: *RecSys*. ACM, 2020, p. 2.

[27] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. “TuckER: Tensor Factorization for Knowledge Graph Completion.” In: *EMNLP/IJCNLP*. 2019.

[28] Tadas Baltrusaitis, Chaitanya Ahuja, and Louis Philippe Morency. “Multimodal Machine Learning: A Survey and Taxonomy.” In: *IEEE Trans. Pattern Anal. Mach. Intell.* 41.2 (2019), pp. 423–443.

[29] Tadas Baltrusaitis, Chaitanya Ahuja, and Louis-Philippe Morency. “Challenges and applications in multimodal machine learning.” In: *The Handbook of Multimodal-Multisensor Interfaces, Volume 2 (2)*. Association for Computing Machinery, 2018, pp. 17–48.

[30] Francesco Barbieri, José Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. “TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification.” In: *EMNLP (Findings)*. Vol. EMNLP 2020. Findings of ACL. Association for Computational Linguistics, 2020, pp. 1644–1650.

[31] Alejandro Bellogín and Alan Said. “Improving accountability in recommender systems research through reproducibility.” In: *User Model. User Adapt. Interact.* 31.5 (2021), pp. 941–977.

[32] Rianne van den Berg, Thomas N. Kipf, and Max Welling. “Graph Convolutional Matrix Completion.” In: *CoRR* abs/1706.02263 (2017).

[33] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. “Algorithms for Hyper-Parameter Optimization.” In: *NIPS*. 2011, pp. 2546–2554.

[34] Gaurav Bhatnagar, Q. M. Jonathan Wu, and Zheng Liu. “Directive Contrast Based Multimodal Medical Image Fusion in NSCT Domain.” In: *IEEE Trans. Multim.* 15.5 (2013), pp. 1014–1024.

[35] Federico Bianchi, Patrick John Chia, Ciro Greco, Claudio Pomo, Gabriel de Souza P. Moreira, Davide Eynard, Fahd Husain, and Jacopo Tagliabue. “EvalRS 2023. Well-Rounded Recommender Systems For Real-World Deployments.” In: *CoRR* abs/2304.07145 (2023).

[36] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. “Evasion Attacks against Machine Learning at Test Time.” In: *ECML-PKDD 2013*. 2013.

[37] Georgios Boltsis and Evaggelia Pitoura. “Bias disparity in graph-based collaborative filtering recommenders.” In: *SAC*. ACM, 2022, pp. 1403–1409.

[38] Ludovico Boratto, Salvatore Carta, Gianni Fenu, and Roberto Saia. “Semantics-aware content-based recommender systems: Design and architecture guidelines.” In: *Neurocomputing* 254 (2017), pp. 79–85.

[39] Ludovico Boratto, Gianni Fenu, and Mirko Marras. “Connecting user and item perspectives in popularity debiasing for collaborative recommendation.” In: *Inf. Process. Manag.* 58.1 (2021), p. 102387.

[40] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. “Translating Embeddings for Modeling Multi-relational Data.” In: *NIPS*. 2013, pp. 2787–2795.

[41] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. “Learning Structured Embeddings of Knowledge Bases.” In: *AAAI*. AAAI Press, 2011.

[42] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. “Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges.” In: *CoRR* abs/2104.13478 (2021).

[43] Erik Brynjolfsson, Yu Jeffrey Hu, and Michael D Smith. “From niches to riches: Anatomy of the long tail.” In: *Sloan management review* 47.4 (2006), pp. 67–71.

[44] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. “nuScenes: A Multimodal Dataset for Autonomous Driving.” In: *CVPR*. Computer Vision Foundation / IEEE, 2020, pp. 11618–11628.

[45] Desheng Cai, Shengsheng Qian, Quan Fang, and Changsheng Xu. “Heterogeneous Hierarchical Feature Aggregation Network for Personalized Micro-Video Recommendation.” In: *IEEE Trans. Multim.* 24 (2022), pp. 805–818.

[46] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. “LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation.” In: *ICLR*. OpenReview.net, 2023.

[47] Jiangxia Cao, Xixun Lin, Shu Guo, Luchen Liu, Tingwen Liu, and Bin Wang. “Bipartite Graph Embedding via Mutual Information Maximization.” In: *WSDM*. ACM, 2021, pp. 635–643.

[48] Nicholas Carlini and David A. Wagner. “Defensive Distillation is Not Robust to Adversarial Examples.” In: *CoRR 2016* (2016).

[49] Nicholas Carlini and David A. Wagner. “Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods.” In: *AISec@CCS 2017*. 2017.

[50] Nicholas Carlini and David A. Wagner. “Towards Evaluating the Robustness of Neural Networks.” In: *SP 2017*. 2017.

[51] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. “Machine Learning on Graphs: A Model and Comprehensive Taxonomy.” In: *J. Mach. Learn. Res.* 23 (2022), 89:1–89:64.

[52] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. “Efficient Neural Matrix Factorization without Sampling for Recommendation.” In: *ACM Trans. Inf. Syst.* 38.2 (2020), 14:1–14:28.

[53] Dapeng Chen, Min Wang, Haobin Chen, Lin Wu, Jing Qin, and Wei Peng. “Cross-Modal Retrieval with Heterogeneous Graph Embedding.” In: *ACM Multimedia.* ACM, 2022, pp. 3291–3300.

[54] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. “Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View.” In: *AAAI.* AAAI Press, 2020, pp. 3438–3445.

[55] Feiyu Chen, Junjie Wang, Yinwei Wei, Hai-Tao Zheng, and Jie Shao. “Breaking Isolation: Multimodal Graph Fusion for Multimedia Recommendation by Edge-wise Modulation.” In: *ACM Multimedia.* ACM, 2022, pp. 385–394.

[56] Hanxiong Chen, Yunqi Li, Shaoyun Shi, Shuchang Liu, He Zhu, and Yongfeng Zhang. “Graph Collaborative Reasoning.” In: *WSDM.* ACM, 2022, pp. 75–84.

[57] Huiyuan Chen and Jing Li. “Neural Tensor Model for Learning Multi-Aspect Factors in Recommender Systems.” In: *IJCAI.* ijcai.org, 2020, pp. 2449–2455.

[58] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. “Bias and Debias in Recommender System: A Survey and Future Directions.” In: *ACM Trans. Inf. Syst.* 41.3 (2023), 67:1–67:39.

[59] Jingjing Chen, Chong-Wah Ngo, Fuli Feng, and Tat-Seng Chua. “Deep Understanding of Cooking Procedure for Cross-modal Recipe Retrieval.” In: *ACM Multimedia.* ACM, 2018, pp. 1020–1028.

[60] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. “Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention.” In: *SIGIR.* ACM, 2017, pp. 335–344.

[61] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. “Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach.” In: *AAAI.* AAAI Press, 2020, pp. 27–34.

[62] Li Chen, Guanliang Chen, and Feng Wang. “Recommender systems based on user reviews: the state of the art.” In: *User Model. User-Adapt. Interact.* (2015).

[63] Tao Chen, Xiangnan He, and Min-Yen Kan. “Context-aware Image Tweet Modelling and Recommendation.” In: *ACM Multimedia.* ACM, 2016, pp. 1018–1027.

[64] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. “On Sampling Strategies for Neural Network-based Collaborative Filtering.” In: *KDD.* ACM, 2017, pp. 767–776.

[65] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. "POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba iFashion." In: *KDD*. ACM, 2019.

[66] Xu Chen, Hanxiong Chen, Hongteng Xu, Yongfeng Zhang, Yixin Cao, Zheng Qin, and Hongyuan Zha. "Personalized Fashion Recommendation with Visual Explanations based on Multimodal Attention Network: Towards Visually Explainable Recommendation." In: *SIGIR*. ACM, 2019, pp. 765–774.

[67] Xusong Chen, Dong Liu, Zhiwei Xiong, and Zheng-Jun Zha. "Learning and Fusing Multiple User Interest Representations for Micro-Video and Movie Recommendations." In: *IEEE Trans. Multim.* 23 (2021), pp. 484–496.

[68] Zhihong Chen, Rong Xiao, Chenliang Li, Gangfeng Ye, Haochuan Sun, and Hongbo Deng. "ESAM: Discriminative Domain Adaptation with Non-Displayed Items to Improve Long-Tail Performance." In: *SIGIR*. ACM, 2020, pp. 579–588.

[69] Zhiyong Cheng, Xiaojun Chang, Lei Zhu, Rose Catherine Kanjirathinkal, and Mohan S. Kankanhalli. "MMALFM: Explainable Recommendation by Leveraging Reviews and Images." In: *ACM Trans. Inf. Syst.* 37.2 (2019), 16:1–16:28.

[70] Zhiyong Cheng, Jialie Shen, and Steven C. H. Hoi. "On Effective Personalized Music Retrieval by Exploring Online User Behaviors." In: *SIGIR*. ACM, 2016, pp. 125–134.

[71] Patrick John Chia, Jacopo Tagliabue, Federico Bianchi, Chloe He, and Brian Ko. "Beyond NDCG: Behavioral Testing of Recommender Systems with RecList." In: *WWW (Companion Volume)*. ACM, 2022, pp. 99–104.

[72] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. "Friendship and mobility: user movement in location-based social networks." In: *KDD*. ACM, 2011, pp. 1082–1090.

[73] Keunwoo Choi, György Fazekas, Mark B. Sandler, and Kyunghyun Cho. "Convolutional recurrent neural networks for music classification." In: *ICASSP*. IEEE, 2017, pp. 2392–2396.

[74] Xiaoya Chong, Qing Li, Howard Leung, Qianhui Men, and Xianjin Chao. "Hierarchical Visual-aware Minimax Ranking Based on Co-purchase Data for Personalized Recommendation." In: *WWW*. ACM / IW3C2, 2020, pp. 2563–2569.

[75] Giandomenico Cornacchia, Vito Walter Anelli, Giovanni Maria Biancofiore, Fedelucio Narducci, Claudio Pomo, Azzurra Ragone, and Eugenio Di Sciascio. "Auditing fairness under unawareness through counterfactual reasoning." In: *Inf. Process. Manag.* 60.2 (2023), p. 103224.

[76] Giandomenico Cornacchia, Francesco M. Donini, Fedelucio Narducci, Claudio Pomo, and Azzurra Ragone. "Explanation in Multi-Stakeholder Recommendation for Enterprise Decision Support Systems." In: *CAiSE Workshops*. Vol. 423. Lecture Notes in Business Information Processing. Springer, 2021, pp. 39–47.

[77] Giandomenico Cornacchia, Fedelucio Narducci, and Azzurra Ragone. “A General Model for Fair and Explainable Recommendation in the Loan Domain (Short paper).” In: *KaRS/ComplexRec@RecSys*. Vol. 2960. CEUR Workshop Proceedings. CEUR-WS.org, 2021.

[78] Qiang Cui, Shu Wu, Qiang Liu, Wen Zhong, and Liang Wang. “MV-RNN: A Multi-View Recurrent Neural Network for Sequential Recommendation.” In: *IEEE Trans. Knowl. Data Eng.* 32.2 (2020), pp. 317–331.

[79] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. “A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research.” In: *ACM Trans. Inf. Syst.* 39.2 (2021), 20:1–20:49.

[80] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. “Are we really making much progress? A worrying analysis of recent neural recommendation approaches.” In: *RecSys*. ACM, 2019, pp. 101–109.

[81] Yashar Deldjoo, Vito Walter Anelli, Hamed Zamani, Alejandro Bellogín, and Tommaso Di Noia. “A flexible framework for evaluating user and item fairness in recommender systems.” In: *User Model. User Adapt. Interact.* 31.3 (2021), pp. 457–511.

[82] Yashar Deldjoo, Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. “A Study on the Relative Importance of Convolutional Neural Networks in Visually-Aware Recommender Systems.” In: *CVPR Workshops*. Computer Vision Foundation / IEEE, 2021, pp. 3961–3967.

[83] Yashar Deldjoo, Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. “Leveraging Content-Style Item Representation for Visual Recommendation.” In: *ECIR (2)*. Vol. 13186. Lecture Notes in Computer Science. Springer, 2022, pp. 84–92.

[84] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. “A Survey on Adversarial Recommender Systems: From Attack/Defense Strategies to Generative Adversarial Networks.” In: *ACM Comput. Surv.* 54.2 (2021), 35:1–35:38.

[85] Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, and Felice Antonio Merra. “How Dataset Characteristics Affect the Robustness of Collaborative Recommendation Models.” In: *SIGIR*. ACM, 2020, pp. 951–960.

[86] Yashar Deldjoo, Markus Schedl, Paolo Cremonesi, and Gabriella Pasi. “Recommender Systems Leveraging Multimedia Content.” In: *ACM Comput. Surv.* 53.5 (2020), 106:1–106:38.

[87] Yashar Deldjoo, Markus Schedl, Balasz Hidasi, Xiangnan He, and Yinwei Wei. “Multimedia Recommender Systems: Algorithms and Challenges.” In: *Recommender Systems Handbook*. Springer US, 2022.

[88] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. “ImageNet: A large-scale hierarchical image database.” In: *CVPR*. IEEE Computer Society, 2009, pp. 248–255.

[89] Marie Destandau and Jean-Daniel Fekete. “The missing path: Analysing incompleteness in knowledge graphs.” In: *Inf. Vis.* 20.1 (2021), pp. 66–82.

[90] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. "Convolutional 2D Knowledge Graph Embeddings." In: *AAAI*. AAAI Press, 2018, pp. 1811–1818.

[91] Xue Dong, Xuemeng Song, Fuli Feng, Peiguang Jing, Xin-Shun Xu, and Liqiang Nie. "Personalized Capsule Wardrobe Creation with Garment and User Modeling." In: *ACM Multimedia*. ACM, 2019, pp. 302–310.

[92] Xiaoyu Du, Zike Wu, Fuli Feng, Xiangnan He, and Jinhui Tang. "Invariant Representation Learning for Multimedia Recommendation." In: *ACM Multimedia*. ACM, 2022, pp. 619–628.

[93] Travis Ebesu, Bin Shen, and Yi Fang. "Collaborative Memory Network for Recommendation Systems." In: *SIGIR*. ACM, 2018, pp. 515–524.

[94] Michael D. Ekstrand, John Riedl, and Joseph A. Konstan. "Collaborative Filtering Recommender Systems." In: *Found. Trends Hum. Comput. Interact.* 4.2 (2011), pp. 175–243.

[95] David Elsweiler, Christoph Trattner, and Morgan Harvey. "Exploiting Food Choice Biases for Healthier Recipe Recommendation." In: *SIGIR*. ACM, 2017, pp. 575–584.

[96] Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. "Graph Trend Filtering Networks for Recommendation." In: *SIGIR*. ACM, 2022, pp. 112–121.

[97] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. "Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO." In: *Semantic Web* 9.1 (2018), pp. 77–129.

[98] Andrea Ferracani, Daniele Pezzatini, Marco Bertini, Saverio Meucci, and Alberto Del Bimbo. "A System for Video Recommendation using Visual Saliency, Crowdsourced and Automatic Annotations." In: *ACM Multimedia*. ACM, 2015.

[99] Zuohui Fu et al. "Fairness-Aware Explainable Recommendation over Knowledge Graphs." In: *SIGIR*. ACM, 2020, pp. 69–78.

[100] Jing Gao, Peng Li, Zhikui Chen, and Jianing Zhang. "A Survey on Deep Learning for Multimodal Data Fusion." In: *Neural Comput.* 32.5 (2020), pp. 829–864.

[101] Jingyue Gao, Yang Lin, Yasha Wang, Xiting Wang, Zhao Yang, Yuanduo He, and Xu Chu. "Set-Sequence-Graph: A Multi-View Approach Towards Exploiting Reviews for Recommendation." In: *CIKM*. ACM, 2020, pp. 395–404.

[102] Xiaoyan Gao, Fuli Feng, Xiangnan He, Heyan Huang, Xinyu Guan, Chong Feng, Zhaoyan Ming, and Tat-Seng Chua. "Hierarchical Attention Network for Visually-Aware Food Recommendation." In: *IEEE Trans. Multim.* 22.6 (2020).

[103] Yunjun Gao, Yuntao Du, Yujia Hu, Lu Chen, Xinjun Zhu, Ziquan Fang, and Baihua Zheng. "Self-Guided Learning to Denoise for Robust Recommendation." In: *SIGIR*. ACM, 2022, pp. 1412–1422.

[104] James Gareth, Witten Daniela, Hastie Trevor, and Tibshirani Robert. *An introduction to statistical learning: with applications in R*. Spinger, 2013.

[105] Yuying Ge, Ruimao Zhang, Xiaogang Wang, Xiaoou Tang, and Ping Luo. "DeepFashion2: A Versatile Benchmark for Detection, Pose Estimation, Segmentation and Re-Identification of Clothing Images." In: *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 5337–5345.

[106] Francesco Gelli, Tiberio Uricchio, Xiangnan He, Alberto Del Bimbo, and Tat-Seng Chua. "Learning Visual Elements of Images for Discovery of Brand Posts." In: *ACM Trans. Multim. Comput. Commun. Appl.* 16.2 (2020), 56:1–56:21.

[107] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. "Semantics-Aware Content-Based Recommender Systems." In: *Recommender Systems Handbook*. Springer, 2015, pp. 119–159.

[108] Marco de Gemmis, Pasquale Lops, and Marco Polignano. "Recommender Systems, Basics of." In: *Encyclopedia of Social Network Analysis and Mining. 2nd Ed.* Springer, 2018.

[109] Mariana-Iuliana Georgescu, Radu Tudor Ionescu, Andreea-Iuliana Miron, Olivian Savencu, Nicolae-Catalin Ristea, Nicolae Verga, and Fahad Shahbaz Khan. "Multimodal Multi-Head Convolutional Attention with Various Kernel Sizes for Medical Image Super-Resolution." In: *WACV*. IEEE, 2023, pp. 2194–2204.

[110] Alireza Gharahighehi and Celine Vens. "Diversification in session-based news recommender systems." In: *Pers. Ubiquitous Comput.* 27.1 (2023), pp. 5–15.

[111] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. "Neural Message Passing for Quantum Chemistry." In: *ICML*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1263–1272.

[112] Xavier Glorot, Antoine Bordes, Jason Weston, and Yoshua Bengio. "A Semantic Matching Energy Function for Learning with Multi-relational Data." In: *ICLR (Workshop Poster)*. 2013.

[113] Kaiqi Gong, Xiao Song, Senzhang Wang, Songsong Liu, and Yong Li. "ITSM-GCN: Informative Training Sample Mining for Graph Convolutional Network-based Collaborative Filtering." In: *CIKM*. ACM, 2022, pp. 614–623.

[114] Alexey Kurakand Ian J. Goodfellow and Samy Bengio. "Adversarial examples the physical world." In: *ICLR 2017*. 2017.

[115] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples." In: *ICLR (Poster)*. 2015.

[116] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Vol. 385. Studies in Computational Intelligence. Springer, 2012.

[117] Aditya Grover and Jure Leskovec. "node2vec: Scalable Feature Learning for Networks." In: *KDD*. ACM, 2016, pp. 855–864.

[118] Asela Gunawardana and Guy Shani. "Evaluating Recommender Systems." In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach, and Bracha Shapira. Springer, 2015, pp. 265–308. DOI: 10.1007/978-1-4899-7637-6_8.

[119] William L. Hamilton. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020.

[120] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S. Davis. "Learning Fashion Compatibility with Bidirectional LSTMs." In: *ACM Multimedia.* ACM, 2017.

[121] F. Maxwell Harper and Joseph A. Konstan. "The MovieLens Datasets: History and Context." In: *ACM Trans. Interact. Intell. Syst.* 5.4 (2016), 19:1–19:19.

[122] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In: *CVPR.* IEEE Computer Society, 2016, pp. 770–778.

[123] Ruining He, Chunbin Lin, Jianguo Wang, and Julian J. McAuley. "Sherlock: Sparse Hierarchical Embeddings for Visually-Aware One-Class Collaborative Filtering." In: *IJCAI.* IJCAI/AAAI Press, 2016, pp. 3740–3746.

[124] Ruining He and Julian J. McAuley. "Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering." In: *WWW.* ACM, 2016, pp. 507–517.

[125] Ruining He and Julian J. McAuley. "VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback." In: *AAAI.* AAAI Press, 2016, pp. 144–150.

[126] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation." In: *SIGIR.* ACM, 2020, pp. 639–648.

[127] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. "Adversarial Personalized Ranking for Recommendation." In: *SIGIR 2018.* 2018.

[128] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. "Neural Collaborative Filtering." In: *WWW.* ACM, 2017, pp. 173–182.

[129] Haithem Hermessi, Olfa Mourali, and Ezzeddine Zagrouba. "Multimodal medical image fusion review: Theoretical background and recent advances." In: *Signal Process.* 183 (2021), p. 108036.

[130] Shawn Hershey et al. "CNN architectures for large-scale audio classification." In: *ICASSP.* IEEE, 2017, pp. 131–135.

[131] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. "Distilling the Knowledge in a Neural Network." In: *CoRR* abs/1503.02531 (2015).

[132] F. L. Hitchcock. "The expression of a tensor or a polyadic as a sum of products." In: *J. Math. Phys* 6.1 (1927), pp. 164–189.

[133] Min Hou, Le Wu, Enhong Chen, Zhi Li, Vincent W. Zheng, and Qi Liu. "Explainable Fashion Recommendation: A Semantic Attribute Region Guided Approach." In: *IJCAI.* ijcai.org, 2019, pp. 4681–4688.

[134] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge J. Belongie, and Deborah Estrin. "Collaborative Metric Learning." In: *WWW.* ACM, 2017, pp. 193–201.

[135] Peng Hu, Liangli Zhen, Dezhong Peng, and Pei Liu. "Scalable Deep Multimodal Learning for Cross-Modal Retrieval." In: *SIGIR.* ACM, 2019, pp. 635–644.

[136] Xuming Hu, Zhijiang Guo, Zhiyang Teng, Irwin King, and Philip S. Yu. "Multimodal Relation Extraction with Cross-Modal Retrieval and Synthesis." In: *ACL (2).* Association for Computational Linguistics, 2023, pp. 303–311.

[137] Yang Hu, Xi Yi, and Larry S. Davis. “Collaborative Fashion Recommendation: A Functional Tensor Factorization Approach.” In: *ACM Multimedia*. 2015.

[138] Chao Huang, Lianghao Xia, Xiang Wang, Xiangnan He, and Dawei Yin. “Self-Supervised Learning for Recommendation.” In: *CIKM*. ACM, 2022, pp. 5136–5139.

[139] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. “MixGCF: An Improved Training Method for Graph Neural Network-based Recommender Systems.” In: *KDD*. ACM, 2021, pp. 665–674.

[140] Prachi Jain, Sushant Rathi, Mausam, and Soumen Chakrabarti. “Knowledge Base Completion: Baseline strikes back (Again).” In: *ArXiv* abs/2005.00804 (2020).

[141] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. “What recommenders recommend: an analysis of recommendation biases and possible countermeasures.” In: *User Model. User Adapt. Interact.* 25.5 (2015), pp. 427–491.

[142] Shatha Jaradat, Nima Dokoohaki, Humberto Jesús Corona Pampín, and Reza Shirvany. “Second Workshop on Recommender Systems in Fashion - fashionXrecsys2020.” In: *RecSys*. ACM, 2020, pp. 632–634.

[143] Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. “A latent factor model for highly multi-relational data.” In: *NIPS*. 2012, pp. 3176–3184.

[144] Xiaowei Jia, Aosen Wang, Xiaoyi Li, Guangxu Xun, Wenyao Xu, and Aidong Zhang. “Multi-modal learning for video recommendation based on mobile application usage.” In: *IEEE BigData*. IEEE Computer Society, 2015, pp. 837–842.

[145] Yiqiao Jin, Yeon-Chang Lee, Kartik Sharma, Meng Ye, Karan Sikka, Ajay Divakaran, and Srijan Kumar. “Predicting Information Pathways Across Online Communities.” In: *CoRR* abs/2306.02259 (2023).

[146] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual Losses for Real-Time Style Transfer and Super-Resolution.” In: *ECCV 2016*. 2016.

[147] Kristiina Jokinen and Topi Hurtig. “User expectations and real experience on a multimodal interactive system.” In: *INTERSPEECH*. ISCA, 2006.

[148] Keller Jordan. “Calibrated Chaos: Variance Between Runs of Neural Network Training is Harmless and Inevitable.” In: *CoRR* abs/2304.01910 (2023).

[149] Marius Kaminskas and Derek Bridge. “Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems.” In: *ACM Trans. Interact. Intell. Syst.* 7.1 (2017), 2:1–2:42.

[150] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. “Correcting Popularity Bias by Enhancing Recommendation Neutrality.” In: *RecSys Posters*. Vol. 1247. CEUR Workshop Proceedings. CEUR-WS.org, 2014.

[151] Wang-Cheng Kang, Chen Fang, Zhaowen Wang, and Julian J. McAuley. “Visually-Aware Fashion Recommendation and Design with Generative Image Models.” In: *ICDM*. IEEE Computer Society, 2017, pp. 207–216.

[152] Atsushi Kawasaki and Akihito Seki. “Multimodal Trajectory Predictions for Autonomous Driving without a Detailed Prior Map.” In: *WACV*. IEEE, 2021, pp. 3722–3731.

[153] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. “Supervised Contrastive Learning.” In: *NeurIPS*. 2020.

[154] Dong Hyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. “Convolutional Matrix Factorization for Document Context-Aware Recommendation.” In: *RecSys*. ACM, 2016, pp. 233–240.

[155] Taeri Kim, Yeon-Chang Lee, Kijung Shin, and Sang-Wook Kim. “MARIO: Modality-Aware Attention and Modality-Preserving Decoders for Multimedia Recommendation.” In: *CIKM*. ACM, 2022, pp. 993–1002.

[156] Yoon Kim. “Convolutional Neural Networks for Sentence Classification.” In: *EMNLP*. ACL, 2014, pp. 1746–1751.

[157] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” In: *ICLR 2015*. 2015.

[158] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks.” In: *ICLR (Poster)*. OpenReview.net, 2017.

[159] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. “Predict then Propagate: Graph Neural Networks meet Personalized PageRank.” In: *ICLR (Poster)*. OpenReview.net, 2019.

[160] Yehuda Koren, Robert M. Bell, and Chris Volinsky. “Matrix Factorization Techniques for Recommender Systems.” In: *Computer* 42.8 (2009), pp. 30–37.

[161] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. “TensorLy: Tensor Learning in Python.” In: *J. Mach. Learn. Res.* 20 (2019), 26:1–26:6.

[162] Dominik Kowald and Emanuel Lacic. “Popularity Bias in Collaborative Filtering-Based Multimedia Recommender Systems.” In: *BIAS*. Vol. 1610. Communications in Computer and Information Science. Springer, 2022, pp. 1–11.

[163] Adit Krishnan, Ashish Sharma, Aravind Sankar, and Hari Sundaram. “An Adversarial Approach to Improve Long-Tail Performance in Neural Collaborative Filtering.” In: *CIKM*. ACM, 2018, pp. 1491–1494.

[164] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” In: *NIPS*. 2012, pp. 1106–1114.

[165] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. “Canonical Tensor Decomposition for Knowledge Base Completion.” In: *ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 2869–2878.

[166] Mounia Lalmas. “Personalising and Diversifying the Listening Experience.” In: *ICTIR*. ACM, 2020, p. 3.

[167] Alfonso Landin, Javier Parapar, and Álvaro Barreiro. “Novel and Diverse Recommendations by Leveraging Linear Models with User and Item Embeddings.” In: *ECIR (2)*. Vol. 12036. Lecture Notes in Computer Science. Springer, 2020, pp. 215–222.

[168] Matthieu Latapy, Clémence Magnien, and Nathalie Del Vecchio. “Basic notions for the analysis of large two-mode networks.” In: *Soc. Networks* 30.1 (2008), pp. 31–48.

[169] Yi-Lun Lee, Yi-Hsuan Tsai, Wei-Chen Chiu, and Chen-Yu Lee. “Multimodal Prompting with Missing Modalities for Visual Recognition.” In: *CVPR*. IEEE, 2023, pp. 14943–14952.

[170] Fei Lei, Zhongqi Cao, Yuning Yang, Yibo Ding, and Cong Zhang. “Learning the User’s Deeper Preferences for Multi-modal Recommendation Systems.” In: *ACM Trans. Multim. Comput. Commun. Appl.* 19.3s (2023), 138:1–138:18.

[171] Zhenfeng Lei, Anwar Ul Haq, Adnan Zeb, Md Suzauddola, and Defu Zhang. “Is the suggested food your desired?: Multi-modal recipe recommendation with demand-based knowledge graph.” In: *Expert Syst. Appl.* 186 (2021), p. 115708.

[172] Cheng-Te Li, Cheng Hsu, and Yang Zhang. “FairSR: Fairness-aware Sequential Recommendation through Multi-Task Learning with Preference Graph Embeddings.” In: *ACM Trans. Intell. Syst. Technol.* 13.1 (2022), 16:1–16:21.

[173] Jiang Li, Xiaoping Wang, Guoqing Lv, and Zhigang Zeng. “GraphMFT: A graph network based multimodal fusion technique for emotion recognition in conversation.” In: *Neurocomputing* 550 (2023), p. 126427.

[174] Jiao Li, Xing Xu, Wei Yu, Fumin Shen, Zuo Cao, Kai Zuo, and Heng Tao Shen. “Hybrid Fusion with Intra- and Cross-Modality Attention for Image-Recipe Retrieval.” In: *SIGIR*. ACM, 2021, pp. 244–254.

[175] Zhan Li, Jinye Peng, Guohua Geng, Xiaojiang Chen, and Pan-Pan Zheng. “Video recommendation based on multi-modal information and multiple kernel.” In: *Multim. Tools Appl.* 74.13 (2015), pp. 4599–4616.

[176] Dawen Liang, Laurent Charlin, James McInerney, and David M. Blei. “Modeling User Exposure in Recommendation.” In: *WWW*. ACM, 2016, pp. 951–961.

[177] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. “Variational Autoencoders for Collaborative Filtering.” In: *WWW*. ACM, 2018, pp. 689–698.

[178] Hongru Liang, Wenqiang Lei, Paul Yaozhu Chan, Zhenglu Yang, Maosong Sun, and Tat-Seng Chua. “PiRhDy: Learning Pitch-, Rhythm-, and Dynamics-aware Embeddings for Symbolic Music.” In: *ACM Multimedia*. ACM, 2020, pp. 574–582.

[179] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. “Improving Graph Collaborative Filtering with Neighborhood-enriched Contrastive Learning.” In: *WWW*. ACM, 2022, pp. 2320–2329.

[180] Fan Liu, Huilin Chen, Zhiyong Cheng, Liqiang Nie, and Mohan S. Kankanhalli. “Semantic-Guided Feature Distillation for Multimodal Recommendation.” In: *CoRR* abs/2308.03113 (2023).

[181] Fan Liu, Zhiyong Cheng, Changchang Sun, Yinglong Wang, Liqiang Nie, and Mohan S. Kankanhalli. "User Diverse Preference Modeling by Multimodal Attentive Metric Learning." In: *ACM Multimedia.* ACM, 2019, pp. 1526–1534.

[182] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. "Interest-aware Message-Passing GCN for Recommendation." In: *WWW.* ACM / IW3C2, 2021, pp. 1296–1305.

[183] Hongtao Liu, Yian Wang, Qiyao Peng, Fangzhao Wu, Lin Gan, Lin Pan, and Pengfei Jiao. "Hybrid neural recommendation with joint deep representation learning of ratings and reviews." In: *Neurocomputing* 374 (2020), pp. 77–85.

[184] Hongtao Liu, Fangzhao Wu, Wenjun Wang, Xianchen Wang, Pengfei Jiao, Chuhan Wu, and Xing Xie. "NRPA: Neural Recommendation with Personalized Attention." In: *SIGIR.* ACM, 2019, pp. 1233–1236.

[185] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, Li Shang, and Ning Gu. "Personalized Graph Signal Processing for Collaborative Filtering." In: *CoRR* abs/2302.02113 (2023).

[186] Kang Liu, Feng Xue, Dan Guo, Peijie Sun, Shengsheng Qian, and Richang Hong. "Multimodal Graph Contrastive Learning for Multimedia-Based Recommendation." In: *IEEE Transactions on Multimedia* (2023), pp. 1–13. DOI: 10.1109/TMM.2023.3251108.

[187] Kang Liu, Feng Xue, Dan Guo, Le Wu, Shujie Li, and Richang Hong. "MEGCF: Multimodal Entity Graph Collaborative Filtering for Personalized Recommendation." In: *ACM Trans. Inf. Syst.* 41.2 (2023), 30:1–30:27.

[188] Qiang Liu, Shu Wu, and Liang Wang. "DeepStyle: Learning User Preferences for Visual Recommendation." In: *SIGIR.* ACM, 2017, pp. 841–844.

[189] Qidong Liu, Jiaxi Hu, Yutian Xiao, Jingtong Gao, and Xiangyu Zhao. "Multimodal Recommender Systems: A Survey." In: *CoRR* abs/2302.03883 (2023).

[190] Siwei Liu, Iadh Ounis, and Craig Macdonald. "An MLP-based Algorithm for Efficient Contrastive Graph Recommendations." In: *SIGIR.* ACM, 2022, pp. 2431–2436.

[191] Wei Liu, Olga Russakovsky, Jia Deng, Fei-Fei Li, and Alex Berg. *ImageNet Large Scale Visual Recognition Challenge 2015.* http://www.image-net.org/challenges/LSVRC/2015/. Accessed: 2021-03-13.

[192] Xiaohao Liu, Zhulin Tao, Jiahong Shao, Lifang Yang, and Xianglin Huang. "EliMRec: Eliminating Single-modal Bias in Multimedia Recommendation." In: *ACM Multimedia.* ACM, 2022, pp. 687–695.

[193] Yong Liu, Susen Yang, Chenyi Lei, Guoxin Wang, Haihong Tang, Juyong Zhang, Aixin Sun, and Chunyan Miao. "Pre-training Graph Transformer with Multimodal Side Information for Recommendation." In: *ACM Multimedia.* ACM, 2021, pp. 2853–2861.

[194] Zhuang Liu, Yunpu Ma, Matthias Schubert, Yuanxin Ouyang, and Zhang Xiong. "Multi-Modal Contrastive Pre-training for Recommendation." In: *ICMR.* ACM, 2022, pp. 99–108.

[195] Zhuoran Liu and Martha A. Larson. “Adversarial Item Promotion: Vulnerabilities at the Core of Top-N Recommenders that Use Images to Address Cold Start.” In: *CoRR* abs/2006.01888 (2020). arXiv: 2006.01888.

[196] Yichao Lu, Ruihai Dong, and Barry Smyth. “Coevolutionary Recommendation Model: Mutual Learning between Ratings and Reviews.” In: *WWW*. ACM, 2018, pp. 773–782.

[197] Sitao Luan, Mingde Zhao, Xiao-Wen Chang, and Doina Precup. “Break the Ceiling: Stronger Multi-scale Deep Graph Convolutional Networks.” In: *NeurIPS*. 2019, pp. 10943–10953.

[198] Fengmao Lv, Xiang Chen, Yanyong Huang, Lixin Duan, and Guosheng Lin. “Progressive Modality Reinforcement for Human Multimodal Emotion Recognition From Unaligned Multimodal Sequences.” In: *CVPR*. Computer Vision Foundation / IEEE, 2021, pp. 2554–2562.

[199] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. “Disentangled Graph Convolutional Networks.” In: *ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 4212–4221.

[200] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. “Learning Disentangled Representations for Recommendation.” In: *NeurIPS*. 2019, pp. 5712–5723.

[201] Mengmeng Ma, Jian Ren, Long Zhao, Davide Testuggine, and Xi Peng. “Are Multimodal Transformers Robust to Missing Modality?” In: *CVPR*. IEEE, 2022, pp. 18156–18165.

[202] Mengmeng Ma, Jian Ren, Long Zhao, Sergey Tulyakov, Cathy Wu, and Xi Peng. “SMIL: Multimodal Learning with Severely Missing Modality.” In: *AAAI*. AAAI Press, 2021, pp. 2302–2310.

[203] Yao Ma, Shilin Zhao, Weixiao Wang, Yaoman Li, and Irwin King. “Multimodality in meta-learning: A comprehensive survey.” In: *Knowl. Based Syst.* 250 (2022), p. 108976.

[204] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. “Towards Deep Learning Models Resistant to Adversarial Attacks.” In: *ICLR 2018*. 2018.

[205] Daniele Malitesta, Giandomenico Cornacchia, Claudio Pomo, Felice Antonio Merra, Tommaso Di Noia, and Eugenio Di Sciascio. “Formalizing Multimedia Recommendation through Multimodal Deep Learning.” In: *CoRR* abs/2309.05273 (2023).

[206] Daniele Malitesta, Giandomenico Cornacchia, Claudio Pomo, and Tommaso Di Noia. “Disentangling the Performance Puzzle of Multimodal-aware Recommender Systems.” In: *EvalRS@KDD*. Vol. 3450. CEUR Workshop Proceedings. CEUR-WS.org, 2023.

[207] Daniele Malitesta, Giandomenico Cornacchia, Claudio Pomo, and Tommaso Di Noia. “On Popularity Bias of Multimodal-Aware Recommender Systems: A Modalities-Driven Analysis.” In: *MMIR@MM*. ACM, 2023, pp. 59–68.

[208] Daniele Malitesta, Giuseppe Gassi, Claudio Pomo, and Tommaso Di Noia. "Ducho: A Unified Framework for the Extraction of Multimodal Features in Recommendation." In: *ACM Multimedia*. ACM, 2023, pp. 9668–9671.

[209] Daniele Malitesta, Claudio Pomo, Vito Walter Anelli, Alberto Carlo Maria Mancino, Eugenio Di Sciascio, and Tommaso Di Noia. "A Topology-aware Analysis of Graph Collaborative Filtering." In: *CoRR* abs/2308.10778 (2023).

[210] Daniele Malitesta, Claudio Pomo, Vito Walter Anelli, Tommaso Di Noia, and Antonio Ferrara. "An Out-of-the-Box Application for Reproducible Graph Collaborative Filtering extending the Elliot Framework." In: *UMAP (Adjunct Publication)*. ACM, 2023, pp. 12–15.

[211] Daniele Malitesta, Claudio Pomo, and Tommaso Di Noia. "Graph Neural Networks for Recommendation: Reproducibility, Graph Topology, and Node Representation." In: *CoRR* abs/2310.11270 (2023).

[212] Alberto Carlo Maria Mancino, Antonio Ferrara, Salvatore Bufi, Daniele Malitesta, Tommaso Di Noia, and Eugenio Di Sciascio. "KGTORe: Tailored Recommendations through Knowledge-aware GNN Models." In: *RecSys*. ACM, 2023, pp. 576–587.

[213] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. "FairMatch: A Graph-based Approach for Improving Aggregate Diversity in Recommender Systems." In: *UMAP*. ACM, 2020, pp. 154–162.

[214] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. "Feedback Loop and Bias Amplification in Recommender Systems." In: *CIKM*. ACM, 2020, pp. 2145–2148.

[215] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. "A Graph-Based Approach for Mitigating Multi-Sided Exposure Bias in Recommender Systems." In: *ACM Trans. Inf. Syst.* 40.2 (2022), 32:1–32:31.

[216] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. "SimpleX: A Simple and Strong Baseline for Collaborative Filtering." In: *CIKM*. ACM, 2021, pp. 1243–1252.

[217] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. "UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation." In: *CIKM*. ACM, 2021, pp. 1253–1262.

[218] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. "Image-Based Recommendations on Styles and Substitutes." In: *SIGIR*. ACM, 2015, pp. 43–52.

[219] . *PyTorch Geometric Documentation. MEMORY-EFFICIENT AGGREGATIONS.* https://pytorch-geometric.readthedocs.io/en/latest/notes/sparse_tensor.html. Accessed online on 15-05-2023. 2022.

[220] Felice Antonio Merra, Vito Walter Anelli, Tommaso Di Noia, Daniele Malitesta, and Alberto Carlo Maria Mancino. "Denoise to Protect: A Method to Robustify Visual Recommenders from Adversaries." In: *SIGIR*. ACM, 2023, pp. 1924–1928.

[221] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. “Efficient Estimation of Word Representations in Vector Space.” In: *ICLR (Workshop Poster)*. 2013.

[222] Weiqing Min, Shuqiang Jiang, and Ramesh C. Jain. “Food Recommendation: Framework, Existing Solutions, and Challenges.” In: *IEEE Trans. Multim.* 22.10 (2020), pp. 2659–2671.

[223] Zongshen Mu, Yueting Zhuang, Jie Tan, Jun Xiao, and Siliang Tang. “Learning Hybrid Behavior Patterns for Multimedia Recommendation.” In: *ACM Multimedia*. ACM, 2022, pp. 376–384.

[224] Mohammadmehdi Naghiaei, Hossein A. Rahmani, and Yashar Deldjoo. “CPFair: Personalized Consumer and Producer Fairness Re-ranking for Recommender Systems.” In: *SIGIR*. ACM, 2022, pp. 770–779.

[225] Vinod Nair and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines.” In: *ICML*. Omnipress, 2010, pp. 807–814.

[226] M. E. J. Newman. “Mixing patterns in networks.” In: *Phys. Rev. E* 67 (2 Feb. 2003), p. 026126. DOI: 10.1103/PhysRevE.67.026126.

[227] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. “Multimodal Deep Learning.” In: *ICML*. Omnipress, 2011, pp. 689–696.

[228] Jianmo Ni, Jiacheng Li, and Julian J. McAuley. “Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects.” In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 188–197.

[229] M. Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. “A Review of Relational Machine Learning for Knowledge Graphs.” In: *Proceedings of the IEEE* 104 (2016), pp. 11–33.

[230] M. Nickel, Volker Tresp, and H. Kriegel. “A Three-Way Model for Collective Learning on Multi-Relational Data.” In: *ICML*. 2011.

[231] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. “A Three-Way Model for Collective Learning on Multi-Relational Data.” In: *ICML*. Omnipress, 2011, pp. 809–816.

[232] Weizhi Nie, Anan Liu, Xiaorong Zhu, and Yuting Su. “Quality models for venue recommendation in location-based social network.” In: *Multim. Tools Appl.* 75.20 (2016), pp. 12521–12534.

[233] Xia Ning, Christian Desrosiers, and George Karypis. “A Comprehensive Survey of Neighborhood-Based Recommendation Methods.” In: *Recommender Systems Handbook*. Springer, 2015, pp. 37–76.

[234] Wei Niu, James Caverlee, and Haokai Lu. “Neural Personalized Ranking for Image Recommendation.” In: *WSDM*. ACM, 2018, pp. 423–431.

[235] Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. “TAaMR: Targeted Adversarial Attack against Multimedia Recommender Systems.” In: *DSN Workshops*. IEEE, 2020, pp. 1–8.

[236] Sergio Oramas, Oriol Nieto, Mohamed Sordo, and Xavier Serra. “A Deep Multimodal Approach for Cold-start Music Recommendation.” In: *DLRS@RecSys*. ACM, 2017, pp. 32–37.

[237] Yi Ouyang, Peng Wu, and Li Pan. “Asymmetrical Context-aware Modulation for Collaborative Filtering Recommendation.” In: *CIKM.* ACM, 2022, pp. 1595–1604.

[238] Charles Packer, Julian J. McAuley, and Arnau Ramisa. “Visually-Aware Personalized Recommendation using Interpretable Image Representations.” In: *CoRR* abs/1806.09820 (2018).

[239] Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. “PinnerSage: Multi-Modal User Embedding Framework for Recommendations at Pinterest.” In: *KDD.* ACM, 2020, pp. 2311–2320.

[240] Dario Di Palma, Vito Walter Anelli, Daniele Malitesta, Vincenzo Paparella, Claudio Pomo, Yashar Deldjoo, and Tommaso Di Noia. “Examining Fairness in Graph-Based Collaborative Filtering: A Consumer and Producer Perspective.” In: *IIR.* Vol. 3448. CEUR Workshop Proceedings. CEUR-WS.org, 2023, pp. 79–84.

[241] Xichen Pan, Peiyu Chen, Yichen Gong, Helong Zhou, Xinbing Wang, and Zhouhan Lin. “Leveraging Unimodal Self-Supervised Learning for Multimodal Audio-Visual Speech Recognition.” In: *ACL (1).* Association for Computational Linguistics, 2022, pp. 4491–4503.

[242] Xingyu Pan, Yushuo Chen, Changxin Tian, Zihan Lin, Jinpeng Wang, He Hu, and Wayne Xin Zhao. “Multimodal Meta-Learning for Cold-Start Sequential Recommendation.” In: *CIKM.* ACM, 2022, pp. 3421–3430.

[243] Vincenzo Paparella. “Pursuing Optimal Trade-Off Solutions in Multi-Objective Recommender Systems.” In: *RecSys.* ACM, 2022, pp. 727–729.

[244] Georgios Paraskevopoulos, Srinivas Parthasarathy, Aparna Khare, and Shiva Sundaram. “Multimodal and Multiresolution Speech Recognition with Transformers.” In: *ACL.* Association for Computational Linguistics, 2020, pp. 2381–2387.

[245] Denis Parra, Antonio Ossa-Guerra, Manuel Cartagena, Patricio Cerda-Mardini, and Felipe del-Rio. “VisRec: A Hands-on Tutorial on Deep Learning for Visual Recommender Systems.” In: *IUI Companion.* ACM, 2021, pp. 5–6.

[246] Bibek Paudel, Fabian Christoffel, Chris Newell, and Abraham Bernstein. “Updatable, Accurate, Diverse, and Scalable Recommendations for Interactive Applications.” In: *ACM Trans. Interact. Intell. Syst.* 7.1 (2017), 1:1–1:34.

[247] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. “Less is More: Reweighting Important Spectral Graph Features for Recommendation.” In: *SIGIR.* ACM, 2022, pp. 1273–1282.

[248] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. “SVD-GCN: A Simplified Graph Convolution Paradigm for Recommendation.” In: *CIKM.* ACM, 2022, pp. 1625–1634.

[249] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. “DeepWalk: online learning of social representations.” In: *KDD.* ACM, 2014, pp. 701–710.

[250] Anne-Flore Nicole Marie Perrin, He Xu, Eleni Kroupi, Martin Rerábek, and Touradj Ebrahimi. “Multimodal Dataset for Assessment of Quality of Experience in Immersive Multimedia.” In: *ACM Multimedia.* ACM, 2015, pp. 1007–1010.

[251] . *PyTorch Geometric Documentation. Creating Message Passing Networks.* https://pytorch-geometric.readthedocs.io/en/latest/notes/create_gnn.html. Accessed online on 15-05-2023. 2022.

[252] Tahleen A. Rahman, Bartlomiej Surma, Michael Backes, and Yang Zhang. "Fairwalk: Towards Fair Graph Embedding." In: *IJCAI.* ijcai.org, 2019, pp. 3289–3295.

[253] Nikhil Rao, Hsiang-Fu Yu, Pradeep Ravikumar, and Inderjit S. Dhillon. "Collaborative Filtering with Graph Information: Consistency and Scalable Methods." In: *NIPS.* 2015, pp. 2107–2115.

[254] Xuan Rao, Lisi Chen, Yong Liu, Shuo Shang, Bin Yao, and Peng Han. "Graph-Flashback Network for Next Location Recommendation." In: *KDD.* ACM, 2022, pp. 1463–1471.

[255] Erzsébet Ravasz and Albert-László Barabási. "Hierarchical organization in complex networks." In: *Phys. Rev. E* 67 (2 Feb. 2003), p. 026112. DOI: 10.1103/PhysRevE.67.026112.

[256] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: *EMNLP/IJCNLP (1).* Association for Computational Linguistics, 2019, pp. 3980–3990.

[257] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: *Conference on Empirical Methods in Natural Language Processing.* 2019.

[258] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. "BPR: Bayesian Personalized Ranking from Implicit Feedback." In: *UAI.* 2009.

[259] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. "GroupLens: An Open Architecture for Collaborative Filtering of Netnews." In: *CSCW.* ACM, 1994, pp. 175–186.

[260] Francesco Ricci, Lior Rokach, and Bracha Shapira, eds. *Recommender Systems Handbook.* Springer, 2015.

[261] Matthew Richardson, Rakesh Agrawal, and Pedro M. Domingos. "Trust Management for the Semantic Web." In: *ISWC.* Vol. 2870. Lecture Notes in Computer Science. Springer, 2003, pp. 351–368.

[262] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification." In: *ICLR.* OpenReview.net, 2020.

[263] Emanuele Rossi, Henry Kenlay, Maria I. Gorinova, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. "On the Unreasonable Effectiveness of Feature Propagation in Learning on Graphs With Missing Node Features." In: *LoG.* Vol. 198. Proceedings of Machine Learning Research. PMLR, 2022, p. 11.

[264] D. Ruffinelli, Samuel Broscheit, and Rainer Gemulla. "You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings." In: *ICLR.* 2020.

[265] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. "You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings." In: *ICLR.* OpenReview.net, 2020.

[266] Noveen Sachdeva, Carole-Jean Wu, and Julian J. McAuley. “On Sampling Collaborative Filtering Datasets.” In: *WSDM.* ACM, 2022, pp. 842–850.

[267] Alan Said and Alejandro Bellogín. “Comparative recommender system evaluation: benchmarking recommendation frameworks.” In: *RecSys.* ACM, 2014, pp. 129–136.

[268] Alan Said and Alejandro Bellogín. “Rival: a toolkit to foster reproducibility in recommender system evaluation.” In: *RecSys.* ACM, 2014, pp. 371–372.

[269] Aghiles Salah, Quoc-Tuan Truong, and Hady W. Lauw. “Cornac: A Comparative Framework for Multimodal Recommender Systems.” In: *J. Mach. Learn. Res.* 21 (2020), 95:1–95:5.

[270] Lei Sang, Min Xu, Shengsheng Qian, Matt Martin, Peter Li, and Xindong Wu. “Context-Dependent Propagating-Based Video Recommendation in Multimodal Heterogeneous Information Networks.” In: *IEEE Trans. Multim.* 23 (2021), pp. 2019–2032.

[271] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. “Item-based collaborative filtering recommendation algorithms.” In: *WWW.* ACM, 2001, pp. 285–295.

[272] . *GitHub. PyG Team. PyTorch_Geometric. Issues. How to make scatter (Just CUDA) results repeatable.* https://github.com/pyg-team/pytorch_geometric/issues/2788. Accessed online on 15-05-2023. 2021.

[273] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. “Modeling Relational Data with Graph Convolutional Networks.” In: *ESWC.* Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 593–607.

[274] Mete Sertkan, Julia Neidhardt, and Hannes Werthner. “PicTouRe - A Picture-Based Tourism Recommender.” In: *RecSys.* ACM, 2020, pp. 597–599.

[275] Ali Shafahi, Mahyar Najibi, AmGhiasi, Zheng Xu, John P. Dickerson, Christoph Studer, Larry S. Davis, GavTaylor, and Tom Goldstein. “Adversarial training for free!” In: *NeurIPS 2019.* 2019.

[276] Guy Shani and Asela Gunawardana. “Evaluating Recommendation Systems.” In: *Recommender Systems Handbook.* Springer, 2011, pp. 257–297.

[277] Tiancheng Shen, Jia Jia, Yan Li, Hanjie Wang, and Bo Chen. “Enhancing Music Recommendation with Social Media Content: an Attentive Multimodal Autoencoder Approach.” In: *IJCNN.* IEEE, 2020, pp. 1–8.

[278] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, Khaled B. Letaief, and Dongsheng Li. “How Powerful is Graph Convolution for Recommendation?” In: *CIKM.* ACM, 2021, pp. 1619–1629.

[279] Liye Shi, Wen Wu, Wenxin Hu, Jie Zhou, Jiayi Chen, Wei Zheng, and Liang He. “DualGCN: An Aspect-Aware Dual Graph Convolutional Network for review-based recommender.” In: *Knowl. Based Syst.* 242 (2022), p. 108359.

[280] Min Shi, Yufei Tang, and Xingquan Zhu. “Topology and Content Co-Alignment Graph Convolutional Learning.” In: *IEEE Trans. Neural Networks Learn. Syst.* 33.12 (2022), pp. 7899–7907.

[281] Juan Shu, Bowei Xi, Yu Li, Fan Wu, Charles A. Kamhoua, and Jianzhu Ma. "Understanding Dropout for Graph Neural Networks." In: *WWW (Companion Volume)*. ACM, 2022, pp. 1128–1138.

[282] Thiago Silveira, Min Zhang, Xiao Lin, Yiqun Liu, and Shaoping Ma. "How good your recommender system is? A survey on evaluations in recommendation." In: *Int. J. Mach. Learn. Cybern.* 10.5 (2019), pp. 813–831.

[283] Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer, and Raquel Urtasun. "Neuroaesthetics in fashion: Modeling the perception of fashionability." In: *CVPR*. IEEE Computer Society, 2015, pp. 869–877.

[284] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *ICLR*. 2015.

[285] Jinbo Song, Chao Chang, Fei Sun, Xinbo Song, and Peng Jiang. "NGAT4Rec: Neighbor-Aware Graph Attention Network For Recommendation." In: *CoRR* abs/2010.12256 (2020).

[286] Mehdi Srifi, Ahmed Oussous, Ayoub Ait Lahcen, and Salma Mouline. "Recommender Systems Based on Collaborative Filtering Using Review Texts - A Survey." In: *Inf.* 11.6 (2020), p. 317.

[287] Dusan Stamenkovic, Alexandros Karatzoglou, Ioannis Arapakis, Xin Xin, and Kleomenis Katevas. "Choosing the Best of Both Worlds: Diverse and Novel Recommendations through Multi-Objective Reinforcement Learning." In: *WSDM*. ACM, 2022, pp. 957–965.

[288] Harald Steck. "Evaluation of recommendations: rating-prediction and ranking." In: *RecSys*. ACM, 2013, pp. 213–220.

[289] Harald Steck. "Embarrassingly Shallow Autoencoders for Sparse Data." In: *WWW*. ACM, 2019, pp. 3251–3257.

[290] Jianing Sun, Zhaoyue Cheng, Saba Zuberi, Felipe Pérez, and Maksims Volkovs. "HGCF: Hyperbolic Graph Convolution Networks for Collaborative Filtering." In: *WWW*. ACM / IW3C2, 2021, pp. 593–601.

[291] Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma, and Mark Coates. "Neighbor Interaction Aware Graph Convolution Networks for Recommendation." In: *SIGIR*. ACM, 2020, pp. 1289–1298.

[292] Jianing Sun et al. "A Framework for Recommending Accurate and Diverse Items Using Bayesian Graph Convolutional Neural Networks." In: *KDD*. ACM, 2020, pp. 2030–2039.

[293] Rui Sun, Xuezhi Cao, Yan Zhao, Junchen Wan, Kun Zhou, Fuzheng Zhang, Zhongyuan Wang, and Kai Zheng. "Multi-modal Knowledge Graphs for Recommender Systems." In: *CIKM*. ACM, 2020, pp. 1405–1414.

[294] Wangbin Sun, Fei Ma, Yang Li, Shao-Lun Huang, Shiguang Ni, and Lin Zhang. "Semi-Supervised Multimodal Image Translation for Missing Modality Imputation." In: *ICASSP*. IEEE, 2021, pp. 4320–4324.

[295] Wenlong Sun, Sami Khenissi, Olfa Nasraoui, and Patrick Shafto. "Debiasing the Human-Recommender System Feedback Loop in Collaborative Filtering." In: *WWW (Companion Volume)*. ACM, 2019, pp. 645–651.

[296] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. “Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison.” In: *RecSys*. ACM, 2020, pp. 23–32.

[297] Anirudh Sundar and Larry Heck. “Multimodal Conversational AI: A Survey of Datasets and Approaches.” In: *ConvAI@ACL*. Association for Computational Linguistics, 2022, pp. 131–147.

[298] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. “Rethinking the Inception Architecture for Computer Vision.” In: *CVPR*. IEEE Computer Society, 2016, pp. 2818–2826.

[299] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. “Intriguing properties of neural networks.” In: *ICLR*. 2014.

[300] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. “A Survey on Deep Transfer Learning.” In: *ICANN (3)*. Vol. 11141. Lecture Notes in Computer Science. Springer, 2018, pp. 270–279.

[301] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. “LINE: Large-scale Information Network Embedding.” In: *WWW*. ACM, 2015, pp. 1067–1077.

[302] Jinhui Tang, Xiaoyu Du, Xiangnan He, Fajie Yuan, Qi Tian, and Tat-Seng Chua. “Adversarial Training Towards Robust Multimedia Recommender System.” In: *IEEE Trans. Knowl. Data Eng.* 32.5 (2020), pp. 855–867.

[303] Wei Tang, Fazhi He, Yu Liu, and Yansong Duan. “MATR: Multimodal Medical Image Fusion via Multiscale Adaptive Transformer.” In: *IEEE Trans. Image Process.* 31 (2022), pp. 5134–5149.

[304] Pongsate Tangseng and Takayuki Okatani. “Toward Explainable Fashion Recommendation.” In: *WACV*. IEEE, 2020, pp. 2142–2151.

[305] Zhulin Tao, Yinwei Wei, Xiang Wang, Xiangnan He, Xianglin Huang, and Tat-Seng Chua. “MGAT: Multimodal Graph Attention Network for Recommendation.” In: *Inf. Process. Manag.* 57.5 (2020), p. 102277.

[306] Nahed Tawfik, Heba A. Elnemr, Mahmoud Fakhr, Moawad I. Dessouky, and Fathi E. Abd El-Samie. “Survey study of multimodality medical image fusion methods.” In: *Multim. Tools Appl.* 80.4 (2021), pp. 6369–6396.

[307] Changxin Tian, Yuexiang Xie, Yaliang Li, Nan Yang, and Wayne Xin Zhao. “Learning to Denoise Unreliable Interactions for Graph Collaborative Filtering.” In: *SIGIR*. ACM, 2022, pp. 122–132.

[308] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. “Complex Embeddings for Simple Link Prediction.” In: *ICML*. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 2071–2080.

[309] Quoc-Tuan Truong, Aghiles Salah, and Hady W. Lauw. “Multi-Modal Recommender Systems: Hands-On Exploration.” In: *RecSys*. ACM, 2021, pp. 834–837.

[310] Saúl Vargas. “Novelty and diversity enhancement and evaluation in recommender systems and information retrieval.” In: *SIGIR*. ACM, 2014, p. 1281.

[311] Saul Vargas and Pablo Castells. “Rank and relevance in novelty and diversity metrics for recommender systems.” In: *RecSys*. ACM, 2011, pp. 109–116.

[312] Kunal Vaswani, Yudhik Agrawal, and Vinoo Alluri. “Multimodal Fusion Based Attentive Networks for Sequential Music Recommendation.” In: *BigMM*. IEEE, 2021, pp. 25–32.

[313] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. “Graph Attention Networks.” In: *ICLR (Poster)*. OpenReview.net, 2018.

[314] Nico Vervliet, Otto Debals, and Lieven De Lathauwer. “Tensorlab 3.0 - Numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization.” In: *ACSSC*. IEEE, 2016, pp. 1733–1738.

[315] Michael Matthias Voit and Heiko Paulheim. “Bias in Knowledge Graphs - An Empirical Study with Movie Recommendation and Different Language Editions of DBpedia.” In: *LDK*. Vol. 93. OASIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 14:1–14:13.

[316] Sanne Vrijenhoek, Mesut Kaya, Nadia Metoui, Judith Möller, Daan Odijk, and Natali Helberger. “Recommenders with a Mission: Assessing Diversity in News Recommendations.” In: *CHIIR*. ACM, 2021, pp. 173–183.

[317] Cheng Wang, Mathias Niepert, and Hui Li. “LRMM: Learning to Recommend with Missing Modalities.” In: *EMNLP*. Association for Computational Linguistics, 2018.

[318] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. “Collaborative Deep Learning for Recommender Systems.” In: *KDD*. ACM, 2015, pp. 1235–1244.

[319] Nan Wang, Lu Lin, Jundong Li, and Hongning Wang. “Unbiased Graph Embedding with Biased Graph Observations.” In: *WWW*. ACM, 2022, pp. 1423–1433.

[320] Qifan Wang, Yinwei Wei, Jianhua Yin, Jianlong Wu, Xuemeng Song, and Liqiang Nie. “DualGNN: Dual Graph Neural Network for Multimedia Recommendation.” In: *IEEE Trans. Multim.* 25 (2023), pp. 1074–1084.

[321] Wenjie Wang, Ling-Yu Duan, Hao Jiang, Peiguang Jing, Xuemeng Song, and Liqiang Nie. “Market2Dish: Health-aware Food Recommendation.” In: *ACM Trans. Multim. Comput. Commun. Appl.* 17.1 (2021), 33:1–33:19.

[322] Wenjie Wang, Fuli Feng, Xiangnan He, Xiang Wang, and Tat-Seng Chua. “Deconfounded Recommendation for Alleviating Bias Amplification.” In: *KDD*. ACM, 2021, pp. 1717–1725.

[323] Xi Wang, Iadh Ounis, and Craig Macdonald. “Leveraging Review Properties for Effective Recommendation.” In: *WWW*. ACM / IW3C2, 2021, pp. 2209–2219.

[324] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. “KGAT: Knowledge Graph Attention Network for Recommendation.” In: *KDD*. ACM, 2019, pp. 950–958.

[325] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. “Neural Graph Collaborative Filtering.” In: *SIGIR*. ACM, 2019, pp. 165–174.

[326] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. “Neural Graph Collaborative Filtering.” In: *CoRR* abs/1905.08108 (2019).

[327] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. “Learning Intents behind Interactions with Knowledge Graph for Recommendation.” In: *WWW*. ACM / IW3C2, 2021, pp. 878–887.

[328] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. “Disentangled Graph Collaborative Filtering.” In: *SIGIR*. ACM, 2020, pp. 1001–1010.

[329] Xiangmeng Wang, Qian Li, Dianer Yu, Peng Cui, Zhichao Wang, and Guandong Xu. “Causal Disentanglement for Semantics-Aware Intent Learning in Recommendation.” In: *CoRR* abs/2202.02576 (2022).

[330] Yifan Wang, Suyao Tang, Yuntong Lei, Weiping Song, Sheng Wang, and Ming Zhang. “DisenHAN: Disentangled Heterogeneous Graph Attention Network for Recommendation.” In: *CIKM*. ACM, 2020, pp. 1605–1614.

[331] Zhenyi Wang, Huan Zhao, and Chuan Shi. “Profiling the Design Space for Graph Neural Networks based Collaborative Filtering.” In: *WSDM*. ACM, 2022, pp. 1109–1119.

[332] Zhidan Wang, Wenwen Ye, Xu Chen, Wenqiang Zhang, Zhenlei Wang, Lixin Zou, and Weidong Liu. “Generative Session-based Recommendation.” In: *WWW*. ACM, 2022, pp. 2227–2235.

[333] Chunyu Wei, Jian Liang, Bing Bai, and Di Liu. “Dynamic Hypergraph Learning for Collaborative Filtering.” In: *CIKM*. ACM, 2022, pp. 2108–2117.

[334] Lanning Wei, Huan Zhao, and Zhiqiang He. “Designing the Topology of Graph Neural Networks: A Novel Feature Fusion Perspective.” In: *WWW*. ACM, 2022, pp. 1381–1391.

[335] Wei Wei, Chao Huang, Lianghao Xia, Yong Xu, Jiashu Zhao, and Dawei Yin. “Contrastive Meta Learning with Behavior Multiplicity for Recommendation.” In: *WSDM*. ACM, 2022, pp. 1120–1128.

[336] Wei Wei, Chao Huang, Lianghao Xia, and Chuxu Zhang. “Multi-Modal Self-Supervised Learning for Recommendation.” In: *WWW*. ACM, 2023, pp. 790–800.

[337] Yinwei Wei, Xiang Wang, Xiangnan He, Liqiang Nie, Yong Rui, and Tat-Seng Chua. “Hierarchical User Intent Graph Network for Multimedia Recommendation.” In: *IEEE Trans. Multim.* 24 (2022), pp. 2701–2712.

[338] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, and Tat-Seng Chua. “Graph-Refined Convolutional Network for Multimedia Recommendation with Implicit Feedback.” In: *ACM Multimedia*. ACM, 2020, pp. 3541–3549.

[339] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. “MMGCN: Multi-modal Graph Convolution Network for Personalized Recommendation of Micro-video.” In: *ACM Multimedia*. ACM, 2019, pp. 1437–1445.

[340] Chuhan Wu, Fangzhao Wu, Tao Qi, Suyu Ge, Yongfeng Huang, and Xing Xie. “Reviews Meet Graphs: Enhancing User and Item Representations for Recommendation with Hierarchical Attentive Graph Neural Network.” In: *EMNLP/I-JCNLP (1)*. Association for Computational Linguistics, 2019, pp. 4883–4892.

[341] Chuhan Wu, Fangzhao Wu, Tao Qi, Chao Zhang, Yongfeng Huang, and Tong Xu. “MM-Rec: Visiolinguistic Model Empowered Multimodal News Recommendation.” In: *SIGIR*. ACM, 2022, pp. 2560–2564.

[342] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. “Simplifying Graph Convolutional Networks.” In: *ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6861–6871.

[343] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. “Self-supervised Graph Learning for Recommendation.” In: *SIGIR*. ACM, 2021, pp. 726–735.

[344] Le Wu, Lei Chen, Richang Hong, Yanjie Fu, Xing Xie, and Meng Wang. “A Hierarchical Attention Model for Social Contextual Image Recommendation.” In: *IEEE Trans. Knowl. Data Eng.* 32.10 (2020), pp. 1854–1867.

[345] Le Wu, Lei Chen, Pengyang Shao, Richang Hong, Xiting Wang, and Meng Wang. “Learning Fair Representations for Recommendation: A Graph-based Perspective.” In: *WWW*. ACM / IW3C2, 2021, pp. 2198–2208.

[346] Qianqian Wu, Pengpeng Zhao, and Zhiming Cui. “Visual and Textual Jointly Enhanced Interpretable Fashion Recommendation.” In: *IEEE Access* (2020).

[347] Jiafeng Xia, Dongsheng Li, Hansu Gu, Jiahao Liu, Tun Lu, and Ning Gu. “FIRE: Fast Incremental Recommendation with Graph Signal Processing.” In: *WWW*. ACM, 2022, pp. 2360–2369.

[348] Lianghao Xia, Chao Huang, Jiao Shi, and Yong Xu. “Graph-less Collaborative Filtering.” In: *CoRR* abs/2303.08537 (2023).

[349] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy X. Huang. “Hypergraph Contrastive Collaborative Filtering.” In: *SIGIR*. ACM, 2022, pp. 70–79.

[350] Yi Xiao, Felipe Codevilla, Akhil Gurram, Onay Urfalioglu, and Antonio M. López. “Multimodal End-to-End Autonomous Driving.” In: *IEEE Trans. Intell. Transp. Syst.* 23.1 (2022), pp. 537–547.

[351] Yongquan Xie, Zhengru Li, Tian Qin, Finn Tseng, Johannes Kristinsson, Shiqi Qiu, and Yi Lu Murphey. “Personalized Session-Based Recommendation Using Graph Attention Networks.” In: *IJCNN*. IEEE, 2021, pp. 1–8.

[352] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. “How Powerful are Graph Neural Networks?” In: *ICLR*. OpenReview.net, 2019.

[353] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. “Representation Learning on Graphs with Jumping Knowledge Networks.” In: *ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 5449–5458.

[354] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. "Embedding Entities and Relations for Learning and Inference in Knowledge Bases." In: *ICLR (Poster)*. 2015.

[355] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. "HOP-rec: high-order proximity for implicit recommendation." In: *RecSys*. ACM, 2018, pp. 140–144.

[356] Liang Yang, Wenmiao Zhou, Weihang Peng, Bingxin Niu, Junhua Gu, Chuan Wang, Xiaochun Cao, and Dongxiao He. "Graph Neural Networks Beyond Compromise Between Attribute and Topology." In: *WWW*. ACM, 2022, pp. 1127–1135.

[357] Longqi Yang, Cheng-Kang Hsieh, Hongjian Yang, John P. Pollak, Nicola Dell, Serge J. Belongie, Curtis Cole, and Deborah Estrin. "Yum-Me: A Personalized Nutrient-Based Meal Recommender System." In: *ACM Trans. Inf. Syst.* 36.1 (2017), 7:1–7:31.

[358] Menglin Yang, Min Zhou, Jiahong Liu, Defu Lian, and Irwin King. "HRCF: Enhancing Collaborative Filtering via Hyperbolic Geometric Regularization." In: *WWW*. ACM, 2022, pp. 2462–2471.

[359] Qi Yang, Gaosheng Wu, Yuhua Li, Ruixuan Li, Xiwu Gu, Huicai Deng, and Junzhuang Wu. "AMNN: Attention-Based Multimodal Neural Network Model for Hashtag Recommendation." In: *IEEE Trans. Comput. Soc. Syst.* 7.3 (2020), pp. 768–779.

[360] Wenzhuo Yang, Jia Li, Chenxi Li, Latrice Barnett, Markus Anderle, Simo Arajärvi, Harshavardhan Utharavalli, Caiming Xiong, and Steven C. H. Hoi. "On the Diversity and Explainability of Recommender Systems: A Practical Framework for Enterprise App Recommendation." In: *CIKM*. ACM, 2021, pp. 4302–4311.

[361] Xun Yang, Xiaoyu Du, and Meng Wang. "Learning to Match on Graph for Fashion Compatibility Modeling." In: *AAAI*. AAAI Press, 2020, pp. 287–294.

[362] Xun Yang, Xiangnan He, Xiang Wang, Yunshan Ma, Fuli Feng, Meng Wang, and Tat-Seng Chua. "Interpretable Fashion Matching with Rich Attributes." In: *SIGIR*. ACM, 2019, pp. 775–784.

[363] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. "Knowledge Graph Contrastive Learning for Recommendation." In: *SIGIR*. ACM, 2022, pp. 1434–1443.

[364] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix X. Yu, Aditya Krishna Menon, Lichan Hong, Ed H. Chi, Steve Tjoa, Jieqi Kang, and Evan Ettinger. "Self-supervised Learning for Deep Models in Recommendations." In: *CoRR* abs/2007.12865 (2020).

[365] Jing Yi and Zhenzhong Chen. "Multi-Modal Variational Graph Auto-Encoder for Recommendation Systems." In: *IEEE Trans. Multim.* 24 (2022), pp. 1067–1079.

[366] Zixuan Yi, Xi Wang, Iadh Ounis, and Craig MacDonald. "Multi-modal Graph Contrastive Learning for Micro-video Recommendation." In: *SIGIR*. ACM, 2022, pp. 1807–1811.

[367] Ruiping Yin, Kan Li, Jie Lu, and Guangquan Zhang. “Enhancing Fashion Recommendation with Visual Compatibility Relationship.” In: *WWW*. ACM, 2019.

[368] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. “A Survey on Multimodal Large Language Models.” In: *CoRR* abs/2306.13549 (2023).

[369] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. “Graph Convolutional Neural Networks for Web-Scale Recommender Systems.” In: *KDD*. ACM, 2018, pp. 974–983.

[370] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. “Self-Supervised Multi-Channel Hypergraph Convolutional Network for Social Recommendation.” In: *WWW*. ACM / IW3C2, 2021, pp. 413–424.

[371] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. “Are Graph Augmentations Necessary?: Simple Graph Contrastive Learning for Recommendation.” In: *SIGIR*. ACM, 2022, pp. 1294–1303.

[372] Penghang Yu, Zhiyi Tan, Guanming Lu, and Bing-Kun Bao. “Multi-View Graph Convolutional Network for Multimedia Recommendation.” In: *CoRR* abs/2308.03588 (2023).

[373] Tong Yu, Yilin Shen, Ruiyi Zhang, Xiangyu Zeng, and Hongxia Jin. “Vision-Language Recommendation via Attribute Augmented Multimodal Reinforcement Learning.” In: *ACM Multimedia*. ACM, 2019, pp. 39–47.

[374] Wenhui Yu, Xiangnan He, Jian Pei, Xu Chen, Li Xiong, Jinfei Liu, and Zheng Qin. “Visually-aware Recommendation with Aesthetic Features.” In: *CoRR* abs/1905.02009 (2019).

[375] Wenhui Yu and Zheng Qin. “Graph Convolutional Network for Recommendation with Low-pass Collaborative Filters.” In: *ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 10936–10945.

[376] Wenhui Yu and Zheng Qin. “Sampler Design for Implicit Feedback Data by Noisy-label Robust Learning.” In: *SIGIR*. ACM, 2020, pp. 861–870.

[377] Xuzheng Yu, Tian Gan, Yinwei Wei, Zhiyong Cheng, and Liqiang Nie. “Personalized Item Recommendation for Second-hand Trading Platform.” In: *ACM Multimedia*. ACM, 2020, pp. 3478–3486.

[378] Jiandian Zeng, Tianyi Liu, and Jiantao Zhou. “Tag-assisted Multimodal Sentiment Analysis under Uncertain Missing Modalities.” In: *SIGIR*. ACM, 2022, pp. 1545–1554.

[379] ChengXiang Zhai, William W. Cohen, and John D. Lafferty. “Beyond independent relevance: methods and evaluation metrics for subtopic retrieval.” In: *SIGIR*. ACM, 2003, pp. 10–17.

[380] Huijing Zhan, Jie Lin, Kenan Emir Ak, Boxin Shi, Ling-Yu Duan, and Alex C. Kot. “A^3-FKG: Attentive Attribute-Aware Fashion Knowledge Graph for Outfit Preference Prediction.” In: *IEEE Trans. Multim.* 24 (2022), pp. 819–831.

[381] Chaohe Zhang, Xu Chu, Liantao Ma, Yinghao Zhu, Yasha Wang, Jiangtao Wang, and Junfeng Zhao. "M3Care: Learning with Missing Modalities in Multimodal Healthcare Data." In: *KDD*. ACM, 2022, pp. 2418–2428.

[382] Jinghao Zhang, Yanqiao Zhu, Qiang Liu, Shu Wu, Shuhui Wang, and Liang Wang. "Mining Latent Structures for Multimedia Recommendation." In: *ACM Multimedia*. ACM, 2021, pp. 3872–3880.

[383] Jinghao Zhang, Yanqiao Zhu, Qiang Liu, Mengqi Zhang, Shu Wu, and Liang Wang. "Latent Structures Mining with Contrastive Modality Fusion for Multimedia Recommendation." In: *CoRR* abs/2111.00678 (2021).

[384] Mengfei Zhang, Cheng Guo, Jiaqi Jin, Mao Pan, and Jinyun Fang. "Sequential Recommendation with Context-Aware Collaborative Graph Attention Networks." In: *IJCNN*. IEEE, 2021, pp. 1–8.

[385] Qi Zhang, Jiawen Wang, Haoran Huang, Xuanjing Huang, and Yeyun Gong. "Hashtag Recommendation for Multimodal Microblog Using Co-Attention Network." In: *IJCAI*. ijcai.org, 2017, pp. 3420–3426.

[386] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric." In: *CVPR 2018*. 2018.

[387] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. "Causal Intervention for Leveraging Popularity Bias in Recommendation." In: *SIGIR*. ACM, 2021, pp. 11–20.

[388] Yiding Zhang, Chaozhuo Li, Xing Xie, Xiao Wang, Chuan Shi, Yuming Liu, Hao Sun, Liangjie Zhang, Weiwei Deng, and Qi Zhang. "Geometric Disentangled Collaborative Filtering." In: *SIGIR*. ACM, 2022, pp. 80–90.

[389] Yin Zhang, Ziwei Zhu, Yun He, and James Caverlee. "Content-Collaborative Disentanglement Representation Learning for Enhanced Recommendation." In: *RecSys*. ACM, 2020, pp. 43–52.

[390] Yiqiao Zhang and Hao Tan. "Effects of Multimodal Warning Types on Driver's Task Performance, Physiological Data and User Experience." In: *HCI (12)*. Vol. 12773. Lecture Notes in Computer Science. Springer, 2021, pp. 304–315.

[391] Yongfeng Zhang and Xu Chen. "Explainable Recommendation: A Survey and New Perspectives." In: *Found. Trends Inf. Retr.* 14.1 (2020), pp. 1–101.

[392] Yuting Zhang, Yiqing Wu, Ran Le, Yongchun Zhu, Fuzhen Zhuang, Ruidong Han, Xiang Li, Wei Lin, Zhulin An, and Yongjun Xu. "Modeling Dual Period-Varying Preferences for Takeaway Recommendation." In: *CoRR* abs/2306.04370 (2023).

[393] Ziqi Zhang, Zeyu Li, Kun Wei, Siduo Pan, and Cheng Deng. "A survey on multimodal-guided visual content synthesis." In: *Neurocomputing* 497 (2022), pp. 110–128.

[394] Lingxiao Zhao and Leman Akoglu. "PairNorm: Tackling Oversmoothing in GNNs." In: *ICLR*. OpenReview.net, 2020.

[395] Minghao Zhao, Le Wu, Yile Liang, Lei Chen, Jian Zhang, Qilin Deng, Kai Wang, Xudong Shen, Tangjie Lv, and Runze Wu. “Investigating Accuracy-Novelty Performance for Graph-based Collaborative Filtering.” In: *SIGIR*. ACM, 2022, pp. 50–59.

[396] Wayne Xin Zhao et al. “RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms.” In: *CIKM*. ACM, 2021, pp. 4653–4664.

[397] Wayne Xin Zhao et al. “RecBole 2.0: Towards a More Up-to-Date Recommendation Library.” In: *CIKM*. ACM, 2022, pp. 4722–4726.

[398] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. “Spectral collaborative filtering.” In: *RecSys*. ACM, 2018, pp. 311–319.

[399] Lei Zheng, Vahid Noroozi, and Philip S. Yu. “Joint Deep Modeling of Users and Items Using Reviews for Recommendation.” In: *WSDM*. ACM, 2017, pp. 425–434.

[400] Tianyue Zheng, Ang Li, Zhe Chen, Hongbo Wang, and Jun Luo. “AutoFed: Heterogeneity-Aware Federated Multimodal Learning for Robust Autonomous Driving.” In: *MobiCom*. ACM, 2023, 15:1–15:15.

[401] Yu Zheng, Chen Gao, Liang Chen, Depeng Jin, and Yong Li. “DGCN: Diversified Recommendation with Graph Convolutional Networks.” In: *WWW*. ACM / IW3C2, 2021, pp. 401–412.

[402] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. “Disentangling User Interest and Conformity for Recommendation with Causal Embedding.” In: *WWW*. ACM / IW3C2, 2021, pp. 2980–2991.

[403] Bolei Zhou, Àgata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. “Learning Deep Features for Scene Recognition using Places Database.” In: *NIPS*. 2014, pp. 487–495.

[404] Hongyu Zhou, Xin Zhou, Zhiwei Zeng, Lingzi Zhang, and Zhiqi Shen. “A Comprehensive Survey on Multimodal Recommender Systems: Taxonomy, Evaluation, and Future Directions.” In: *CoRR* abs/2302.04473 (2023).

[405] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. “Towards Deeper Graph Neural Networks with Differentiable Group Normalization.” In: *NeurIPS*. 2020.

[406] Xin Zhou and Zhiqi Shen. “A Tale of Two Graphs: Freezing and Denoising Graph Structures for Multimodal Recommendation.” In: *CoRR* abs/2211.06924 (2022).

[407] Xin Zhou, Hongyu Zhou, Yong Liu, Zhiwei Zeng, Chunyan Miao, Pengwei Wang, Yuan You, and Feijun Jiang. “Bootstrap Latent Representations for Multi-modal Recommendation.” In: *WWW*. ACM, 2023, pp. 845–854.

[408] Yuchen Zhou, Yanan Cao, Yanmin Shang, Chuan Zhou, Shirui Pan, Zheng Lin, and Qian Li. “Explainable Hyperbolic Temporal Point Process for User-Item Interaction Sequence Generation.” In: *ACM Trans. Inf. Syst.* 41.4 (2023), 83:1–83:26.

[409] Jieming Zhu, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Xi Xiao, and Rui Zhang. "BARS: Towards Open Benchmarking for Recommender Systems." In: *SIGIR*. ACM, 2022, pp. 2912–2923.

[410] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Yuanqi Du, Jieyu Zhang, Qiang Liu, Carl Yang, and Shu Wu. "A Survey on Graph Structure Learning: Progress and Opportunities." In: *CoRR* abs/2103.03036 (2022).

[411] Ziwei Zhu, Jianling Wang, and James Caverlee. "Measuring and Mitigating Item Under-Recommendation Bias in Personalized Ranking Systems." In: *SIGIR*. ACM, 2020, pp. 449–458.

[412] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. "Improving recommendation lists through topic diversification." In: *WWW*. ACM, 2005, pp. 22–32.

[413] Qin Zou, Zheng Zhang, Qian Wang, Qingquan Li, Long Chen, and Song Wang. "Who Leads the Clothing Fashion: Style, Color, or Texture? A Computational Study." In: *CoRR* abs/1608.07444 (2016).